THE UNIVERSITY OF ALBERTA

RELEASE FORM

NAME OF AUTHOR ....D. Bruce Butler........

TITLE OF THESIS ....Computerized Manpower Scheduling........

....

....

DEGREE FOR WHICH THESIS WAS PRESENTED ....M. Sc.........

YEAR THIS DEGREE GRANTED ....1978........

Permission is hereby granted to THE UNIVERSITY OF ALBERTA LIBRARY to reproduce single copies of this thesis and to lend or sell such copies for private, scholarly or scientific research purposes only.

The author reserves other publication rights, and neither the thesis nor extensive extracts from it may be printed or otherwise reproduced without the author's written permission.

DATED ...Sept

THE UNIVERSITY OF ALBERTA

Computerized Manpower Scheduling

by

© D. B. Butler

A THESIS

SUBMITTED TO THE FACULTY OF GRADUATE STUDIES AND RESEARCH

IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE

OF Master of Science

IN

Management Science

Department of Computing Science

Department of Finance and Management Science

EDMONTON, ALBERTA

FALL 1978

Abstract

A new Three Stage Method for generating cyclic, shift schedules is described. This method differs from existing methods described in the literature in two significant ways. It contains a procedure to explicitly assign the shifts to be worked and the method is a more general, enumerative, as opposed to heuristic, procedure. The method is shown to be capable of solving real problems and compares favourably with other methods. Alternative algorithms, which adopt different objectives for the assignment of shifts, the third stage, are described and compared. Finally, possible approaches for extending this research are discussed.

Acknowledgements

I would like to express my appreciation to Ursula Maydell, Terry Daniel and the other faculty members who supported my interdisciplinary program and assisted me with this dissertation, despite that it was not directly related to their current research interests. I would also like to acknowledge Robert F. Lunney, Chief of Police, for fostering an environment compatible with research and innovation, my colleague Bill Brown for giving me encouragement when progress was slow, and the various members of the Edmonton Police Department for their cooperation. In addition, I would like thank Jerry Juzwa for drawing the diagrams and Linda Jacox for typing the special symbols.

Table of Contents

List of Tables

List of Figures

CHAPTER 1 Introduction and Overview of Manpower Scheduling

1.1 Introduction

We are living at a time when shift work is commonplace. Emergency services such as police, fire and ambulance service work seven days a week and twenty-four hours a day due to public need; capital intensive industries such as manufacturing plants, petroleum refineries and large computer installations operate for more than the standard five day, one shift week in order to remain economically viable. As facility costs such as office rental, furnishings and equipment consume a larger share of expenditures, the trend will be towards a greater proportion of the workforce to be on shift work[1]. It will soon be uneconomical to have office building complexes in use for only a fraction of the time. Industries that presently operate on a one shift, five day week will have to consider expanding their hours of operation while those already utilizing shift work, especially public sector service agencies, will have to make more judicious use of their manpower and equipment in order to increase productivity.

----

[1] There is no data available on the present proportion of the Canadian workforce on shift work. However, according to Knauth et al. [31] and Rutenfranz et al. [43], approximately 16 to 22 per cent of the workforces of several European countries are on shift work. It should be noted that the definition of shift work might vary by country but Knauth et al. state that the proportion is increasing in these countries.

Because of the economic considerations and the volume of people affected, sophisticated automated techniques need to be employed for manpower scheduling; the administrator or manager no longer can afford to use ad hoc, manually devised scheduling arrangements. As management becomes more cognizant of the human and occupational health factors connected with manpower scheduling, either voluntarily or through labour contractual obligations, the scheduling task becomes increasingly complex. These ergonomic aspects of shift work have lately come under much study, mostly in Europe [38, 30, 31, 40, 41, 43].

This study of manpower scheduling was precipitated by the needs of the City of Edmonton Police Department in Alberta, Canada. Since a review of the literature did not reveal a method suitable for use by the Police Department, a new method was developed. This literature review comprises Chapter 2 while the new method is described in Chapter 3. Chapter 4 consists of a discussion of the computational characteristics of the new method and a comparison with an existing technique. Details of the application of the newly developed method in the Edmonton Police Department are given in Chapter 5. Chapter 6 concludes the dissertation by outlining areas of future research.

1.2 Overview

Manpower scheduling is defined to be the process of determining the temporal distribution of the workforce. This can be separated into two largely distinct problems:

- <u>manpower allocation</u> - the allocation of the manpower to various times of the for each day of some period of days (usually a week). This allocation is accomplished by having several different time periods of work, called shifts. For example, twenty-four hours could be divided into three shifts: Nights (Midnight - 8 AM), Days (8 AM - 4 PM) and Afternoons (4 PM - Midnight). Often the term shift refers to more than one period of working hours. For example, to provide overlapping coverage, an organization might schedule employees from 7 AM to 3 PM and from 8 AM to 4 PM but refer to both time periods as Day shift. An example of a manpower allocation is given in Figure 1.
- <u>shift scheduling</u> - the assignment of groups of employees (ie. one or more persons) to work in specific shifts for specific days. This requires the generation of shift schedules or duty rosters. A shift schedule outlines the shifts to be worked and the days off for the respective groups of employees.

These two problems are explained below.

Each entry refers to the "number of groups of employees" either assigned to a shift (for two shifts - Days and Afternoons) or having the day off for the particular day of the week. This allocation is for 5 groups of people. As an example, on Mondays, 2 groups would be working Day shift, 1 group working Afternoon shift and 2 groups having the day off.

	Mon	Tue	Wed	Thu	Fri	Sat	Sun
Days - D	2	2	2	2	2	2	2
Afternoons - A	1	1	2	2	2	1	0
Day off	2	2	1	1	1	2	3

FIGURE 1 - Example of a Manpower Allocation

1.2.1 Manpower Allocation

The solution to the manpower allocation problem can be deterministic in number such as when scheduling a manufacturing concern that requires a specific number of workers at all times, or variable in number, such as a senior nurse's estimation of the number of nurses needed during a shift. The degree of sophistication in allocation procedures varies widely. Many public service agencies, for simplicity, use a naive equal assignment of manpower to all shifts regardless of workload. Some organizations allocate manpower proportional to some measure of workload. For example, if twice as many calls for police assistance occur during afternoon shift as compared to night shift, twice as much manpower would be assigned to the afternoon shift as compared to the night shift. Still other organizations utilize dynamic techniques that optimize a level of service according to specified criteria and incorporate demand forecasting using such tools as time series analysis and queueing models. For example, Kolesar et al. [32] utilize an M/M/n (Poisson arrival of calls, exponential service times per call with n servers) queueing model using historical data to estimate the minimum number of patrol cars, n, needed in each hour such that the probablity of all cars being busy (ie. probability of a call being queued) is less than some specified level. These estimates are then used in an integer program to specify the number of patrol cars, and

police officers, required on each tour of duty. Irrespective of the method used, objectives are constrained by physical limitations, such as availability of patrol vehicles, restrictions imposed by labour contracts, such as a specified maximum percentage of time worked on any one shift, and traditional practices, such as shift starting times.

The result of all allocation procedures is a matrix of the number of groups of workers required for each shift for each day and the number of groups to be off each day. Often the manpower allocation is assumed to be the same for all weeks so that the resulting matrix gives the number of groups required for each shift for each day of the week. A group is a collection of workers such that each person in the group works the same schedule, with the same shift and days off assignment. For all the shift scheduling methods discussed in this thesis, these groups should contain approximately the same number of people. This matrix then is input either as a constraint or as an objective into the shift scheduling problem.

1.2.2 Shift Scheduling

The solutions to the shift scheduling problem can be considered as principally the concern of the workers. The objective of this problem should be to maximize the satisfaction of the workers with respect to manpower scheduling while adhering to management's specification of manpower allocation. The optimization of the employees' satisfaction is a difficult task due to the multiplicity of interrelated desires and their priorities. For example, these could be measured by the frequency of weekends off, the average length of working periods, and the frequency of shift changes.

Shift scheduling terminology is not used consistently in the literature. Therefore the descriptions of shift schedules that are used in this thesis are defined below.

A cyclic shift schedule consists of a sequence of weeks with the number of weeks[2] equal to the number of groups of workers to be scheduled. At any point in time, each group is working a different week in the schedule. The groups work through the sequence of weeks and at the end of the last week cycle back to the beginning of the first week in the schedule. Thus all groups of employees have the same sequence of working specific shifts and having specific days-off periods. However, different groups start the cycle

---

[2] The term 'week' is not restricted to the calendar week of seven days but unless specified otherwise is assumed to be a seven day period.

at different time points (ie. different weeks). An example of a cyclic shift schedule is contained in Figure 2.

A period of successive working days is termed a _working period_, or WP, while a period of successive days off is termed a _days-off period_, or DOP. A rotation is a sequence of days on and off but with no shifts specified. A _rotation_ can be uniquely described by specifying the sequence of days-off periods and the lengths of either the preceding or following working periods.

Individual schedules are those where each group works a schedule different than those of other groups. An example is shown in Figure 3. Individual scheduling thus can deal with preferences of individual groups (or individuals if there is one person per group). In addition, the manpower allocation can change over time. However, the the generation of schedules is more complex because as many schedules must be produced as there are groups. The problem can be simplified somewhat if the individual schedules can be developed for a short period of time and repeated. This will then only allow the allocation to vary within the period. A special case of this type of schedule is the _fixed schedule_ where each group works the same days on and off every week. An example of a fixed schedule is shown in Figure 4.

Both cyclic and individual schedules can have either a _rotational shift assignment_, where the workers rotate through the various shifts, or a _fixed shift assignment_. It is also possible to schedule a portion of the workforce

This shift schedule meets the manpower allocation for 5 groups of employees in Figure 1. Groups of employees work sequentially throught the weeks and at the end of week #5 cycle back to the beginning of week #1. Each of the 5 groups start the cycle with a different week (ie. the first group starts with week #1, the second group with week #2, etc.).

Week #	Mon	Tue	Wed	Thu	Fri	Sat	Sun
1	*	*	A	A	A	A	*
2	A	A	A	A	A	*	*
3	D	D	D	D	*	*	*
4	D	D	D	*	D	D	D
5	*	*	*	D	D	D	D

Legend:

* - Day off

D - Day Shift

A - Afternoon Shift

FIGURE 2 - Example of a Cyclic Shift Schedule

Individual schedules for seven employees for a two-week period are shown below. A new set of schedules would be used beyond this period.

Employee #	M	T	W	T	F	S	S	M	T	W	T	F	S	S
1	*	*	*	D	D	D	D	*	A	A	A	A	*	*
2	D	D	D	A	A	*	*	D	D	D	D	D	*	*
3	D	D	D	D	*	*	*	A	A	A	A	*	D	D
4	D	*	*	A	A	*	*	A	A	A	A	A	A	A
5	A	A	A	A	A	A	A	*	*	D	D	D	*	*
6	A	A	A	*	D	D	D	D	D	D	D	*	*	*
7	*	*	D	D	D	*	*	D	D	*	*	D	D	D

Legend: * - Day off

D - Day shift

A - Afternoon shift

FIGURE 3 - Example of an Individual Schedule

Each of the six employees would work the same days and have the same days off every week. If the working days were always the same shift, this schedule would have a fixed shift assignment.

Employee #	Mon	Tue	Wed	Thu	Fri	Sat	Sun
1						*	*
2						*	*
3				*	*		
4	*	*					
5		*	*				
6			*	*			

Legend: * - Day off

FIGURE 4 - Example of Fixed Schedule

using one type of scheduling and the remainder of the workforce using another by splitting up the allocation and generating the shift schedules for the two portions separately.

As first demonstrated by Bodin [10], a procedure for generating cyclic schedules can be divided into three stages:

1. Determination of days-off periods

   From the distribution of days off specified in the manpower allocation, this stage groups them into days-off periods, also referred to as recreation clusters. Workers' preferences are incorporated by assigning a weight to each acceptable days-off period and from this set a combination of days-off periods that maximizes the sum of the weights and satisfies the allocation is chosen. The relative values of these weights would reflect the workers' preferences. Constraints specifying minimum and maximum frequencies of days-off periods can be incorporated in this stage.

2. Generation of rotations of days-off periods

   This stage takes the days-off periods determined in stage 1 and determines their order in the schedule. At this stage, constraints specifying allowable lengths of working periods and restrictions on the rotation of the days-off periods, such as weekends off not being allowed

to be consecutive, can be included. This stage produces sequences of days worked and days off.

3. Assignment of shifts to working days

This stage takes the sequences produced by stage 2 and specifies the shifts for the days worked, where constraints restricting the shift assignment, such as minimum time off between shifts, must be imposed. For schedules with fixed shift assignments, this stage is trivial because there is only one shift.

The newly developed Three Stage Method, described in Chapter 3, is based on these stages. For an example illustrating the three stages, the reader is referred to Figures 16, 17 and 18 in Chapter 3. Procedures described in the literature review of Chapter 2 will be discussed in the context of the above three stages.

The methcdology of the first two stages of the Three Stage Method is similar to methods devised by Heller et al. [22, 23, 24] and Bodin [10] but is of a more general nature. The third stage, which has been the major thrust of the research, does not resemble any previously developed procedure. The specification of shifts to be worked has rarely been dealt with in the literature and the methods presented have not been well defined algorithms. Thus this third stage is the most significant contribution of this Three Stage Method.

1.3 Specific Application

This work was initiated by a problem with scheduling police officers in the Edmonton Police Department. Nine groups of officers were required to work three different shifts, with more groups on duty on Afternoon shift than on the others. It was realized that a cyclic shift schedule, satisfying constraints with respect to days-off periods, working period lengths and shift changes, needed to be determined. Since a suitable method for generating such schedules did not exist, a general method, called the Three Stage Method, was developed (see Chapter 3). This new procedure produced an acceptable schedule for the patrol officers of the Police Department. Should the scheduling requirements change, this new systematic scheduling method will facilitate the generation of new and different shift schedules.

CHAPTER 2 Review of Literature

2.1 Classification of Scheduling Methods

The importance of manpower scheduling can be inferred by the wide range of applications discussed in the literature. The scheduling of nursing personnel appears to be the topic of the most research in this area [3, 18, 20, 25, 37, 38, 39, 41, 46, 48] while other occupations whose problems are discussed are transit operators [8, 19, 35], police officers [13, 16, 22, 23, 24], telephone operators [12, 14, 36], garbage collectors [10] and baggage handlers [40]. Bodin [10] describes a general model for manpower scheduling in which he imbeds four procedures in use by four different types of occupations. Several authors [1, 5, 21, 44, 47] discuss the general aspects of scheduling that are not specific to any one industry. The scheduling problem in certain areas of the transportation industry, such as for transit operators and airline crews, is usually much more complex than the general manpower scheduling problem because one has to be concerned with the spatial distribution, especially in airline crew scheduling, in addition to the temporal distribution. Knowing when people are scheduled to work is not sufficient; one has to determine where they are to be scheduled and ensure that it is feasible to travel from one location to another.

In order to compare various shift scheduling procedures, it is useful to categorize them according to the type of the methods and the schedules produced. The two-way classification used by Smith [46] differentiates between the type of schedule (cylical or non-cyclical) and the solution procedure (heuristic and optimizing). Smith also included some procedures that do not generate shift schedules but were only concerned with the allocation of manpower. The distinction between cyclic and individual schedules will be retained here but the nature of the solution procedure would be better represented by a continuum ranging from trial and error on one hand to non-heuristic, or completely optimizing, on the other.

In this thesis the procedures are grouped into three broad categories: trial and error, heuristic, and enumerative. <u>Trial and error</u> is loosely defined to include those procedures that incorporate the intuition of the human scheduler in performing some part of the first two stages.[3] These include those methods where the first stage might be solved by a linear program but the second stage relies on the scheduler. <u>Enumerative procedures</u> are defined to be those where schedules are generated systematically for combinations of various factors such as working periods and days-off periods. <u>Heuristic procedures</u> may then be defined as the generation of only one schedule sqatisfying specific

[3] As several methods do not address the problem of the assignment of shifts worked, the procedure used for the third stage is not included as a criterion in the classification.

conditions; however this schedule is not necessarily an optimal one and improvement thereof can only be done by trial and error. Various authors referred to in this literature review have implicitly used this definition of heuristic procedure. Perhaps it should be noted that the connotation of the term "heuristic", as used in computer science today, is not the same as the definition above.

It also could be desirable to create a third dimension regarding the level of inclusion of workers' preferences. This dimension could be broken into three classes: not included, some implicitly accounted for (eg. a spreading equation used to space weekends), or explicitly included as in an objective function.* Due to the small number of procedures that utilize workers' preferences, this component has not been included in the classification scheme; nevertheless, the inclusion of workers' preferences is discussed where appropriate. Additionally, one could classify according to whether a procedure is used manually or implemented on a computer. However, it was found that several of the manual methods are well enough defined to be considered algorithms which could be computerized, whereas some of the computerized methods are little more than book-keeping devices for trial and error procedures. Hence, the references pertaining to shift scheduling procedures may be classed according to trial and error, heuristic and

----

* The distinction is made between constraints set down by the workers (ie. binding) and their preferences (ie. objectives).

enumerative approaches, and whether the schedules are cyclic or individual, as given below.

Cyclic Schedules

Trial and Error Procedures

Dalley [16]

Frances [18]

Healy [21]

Howell [25]

Maier-Rothe and Wolfe [37]

Monroe [40]

Rothstein [44] [5]

Heuristic Procedures

Bennett and Potts [8]

Bodin [10]

Guha and Browne [19]

Kregeloh and Miodrag [35]

Smith [46]

---

[5] This procedure consists of an optimization for stage 1 but no well defined procedure for stage 2.

Enumerative Procedures

Heller [22] with Heller, McEwen and Stenzel [23, 24]

Individual Schedules

Trial and Error Procedures

Ahuja and Sheppard [3]

Morrish and O'Connor [41] [6]

Heuristic Procedures

Baker and Magazine [7]

Buffa et al. [12] with Luce [36]

Enumerative Procedures

Miller [39]

---

[6] In Ahuja and Sheppard [3] and Morrish and O'Connor [41] the days off are cyclic but the shift assignments are individual.

2.2 Methods for Cyclic Schedules

2.2.1 Trial and Error Procedures

The original work in systematic approaches to shift scheduling appears to be that of Healy [21] as applied to round-the-clock manufacturing operations. Healy introduces the notion that the simplest approach to developing rotational schedules is to have the length of the schedule (in weeks) equal to the number of groups of employees being scheduled. His procedures developing these schedules are trial and error.

Monroe [40] extends Healy's work but deals with restricted, somewhat ideal problems. He starts by adding the constraint that all days off (called regular days off or RDO's in the paper) are to be consecutive. He gives the set of linear equations, shown in Figure 5, that when solved yield the number of each consecutive day off pair. Rather than obtaining the solution by solving these equations, he employs the following algorithm. Select a starting number of Monday-Tuesday pairs (he suggests about half of the number of RDO's for Tuesday). Subtract this number from the number of RDO's for Tuesday. Assign this difference as the number of Tuesday-Wednesday pairs and continue this process until calculating the number of Sunday-Monday pairs. Subtract this number from the number of Monday RDO's. If this difference

The following equations are solved:

$$
\begin{array}{ccccccccccccccc}
MT & & TW & & WT & & TF & & FS & & SS & & SM & & \\
X_1 & + & X_2 & & & & & & & & & & & = & Y_1 \\
 & & X_2 & + & X_3 & & & & & & & & & = & Y_2 \\
 & & & & X_3 & + & X_4 & & & & & & & = & Y_3 \\
 & & & & & & X_4 & + & X_5 & & & & & = & Y_4 \\
 & & & & & & & & X_5 & + & X_6 & & & = & Y_5 \\
 & & & & & & & & & & X_6 & + & X_7 & = & Y_6 \\
X_1 & + & & & & & & & & & & & X_7 & = & Y_7
\end{array}
$$

where X_1 = number of Mon-Tue days-off periods

X_2 = number of Tue-Wed days-off periods, etc.

Y_1 = frequency of days off on Tuesday

Y_2 = frequency of days off on Wednesday, etc.

and $X_1, X_2, \ldots$ are integers

FIGURE 5 - Monroe's Equations

equals the original number chosen for Monday-Tuesday pairs, the solution has been determined. If not, take the average of both the original and later values, use this as the number of Monday-Tuesday pairs and repeat the above process. A solution is guaranteed this second time and it is feasible if all the numbers of pairs are non-negative. If the solution is not feasible, Monroe suggests either altering the allocation, by changing requirements or adding personnel, or by scheduling non-consecutive days off. Monroe supplies a heuristic approach for performing the latter.

For developing the rotation of these days off, Monroe starts the schedule with all the Monday-Tuesday pairs, then all the Tuesday-Wednesday pairs, and continues advancing until all pairs have been included. He then attempts to assign the shifts in such a manner that there is no change of shift during a working period. If this is not possible but desirable, he then changes the rotation of the day off pairs and attempts to assign the shifts again. Monroe does not describe any algorithms to generate these rotations of days-off pairs and shift assignments and only suggests a manual, trial and error approach. He does not accomodate any constraints regarding minimum and maximum lengths of working periods but he does ensure that there are five days worked and two days off for every week (defined as Monday to Sunday).

The application of Monroe's procedures to the hospital nursing shift scheduling problem is described in Howell [25]

and Frances [18] with the latter article discussing the problems associated with implementation of the schedules produced.

Rothstein [44] describes an integer programming formulation of the first stage, the determination of the days-off periods. He adapted this from Monroe [40] for the case where non-consecutive days off are allowed yet there are two days off each week. The objective function is to minimize the number of non-consecutive pairs. The optimal solution, to the linear program which is shown in Figure 6, will always consist of integers. The first seven constraints ensure that all the days off for each day of the week are assigned; the eighth constraint determines d, the number of non-consecutive pairs[7]; and the last seven constraints ensure the ability to group the single days off into feasible pairs. A non-feasible pair is two of the same day; they both can't be assigned to the same week. It is also possible to add other constraints such as minimum number of Saturday-Sunday pairs. No procedure for developing the rotation of the days-cff periods was given and since Rothstein's application was to workers on single shift, the third stage is trivial.

Maier-Rothe and Wolfe [37] describe an approach to generating shift schedules which inccrporate different

----

[7] The variable d and this eighth constraint can be eliminated by stating the objective function as maximizing the number of consecutive pairs. ie. Maximize $z = x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7$.

Minimize d

subject to

$$x_i + x_{i+1} + u_i = b_i, \ i = 1,\ldots,6$$

$$x_1 + x_7 \quad + u_7 = b_7$$

$$x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 + d \le \frac{1}{2} \sum_{i=1}^{7} u_i$$

$$u_i \le \sum_{j \ne i} u_j, \ i = 1,\ldots,7$$

where

x_i, $i = 1,\ldots,7$ is # of days-off pairs of MT, TW, ..., SM

u_i, $i = 1,\ldots,7$ is # of single days off of T, W, ..., M

b_i, $i = 1,\ldots,7$ is total # of days off of T, W, ..., M

d is total # of single days off

FIGURE 6 - Rothstein's Equations

levels of staff[8] and shift to shift adjustments to compensate for variations in workload and staff absences. To determine the days-off periods, they describe the constraints which would ensure that the maximum number of consecutive working days was not exceeded. However they state that the days-off are determined "more easily" by trial and error but they do not describe the procedure they used for this stage. Also they did not discuss the development of a rotation of these days-off periods. As they were concerned with single shift applications, the assignment of shifts to working days was not relevant.

Dalley [16] describes manual procedures for determining hours of shifts, the allocation of manpower to the shifts and the generation of shift schedules. Only the latter procedure will be described in this thesis. From the manpower allocation, Dalley generates shift schedules by a somewhat arbitrary method. He first selects the day of the week that is off the fewest number of times (but non-zero) and calculates during what shift(s) this day should be off by subtracting the number of times this day is worked from the number of weeks assigned for each shift. Of the possible weeks in the schedule from which to choose, Dalley states that the choice can be made arbitrarily. He states no reason why the day off with lowest frequency is scheduled first and if there are several days of equal frequency, the day chosen from among them is arbitrary. This above procedure is

---

[8] ie. Registered nurses (RN's), Licensed practical nurses (LPN's) and Registered nursing aides (RNA's).

repeated until all days off have been scheduled. No regard appears to have been taken with respect to factors such as consecutive days off or length of working and day off periods; if a problem does occur, he suggests taking certain days off in compensation for a statutory holiday or accumulated overtime. He also suggests that if the schedule produced is not satisfactory, it "can be re-arranged an unlimited number of times" to make an acceptable one but he does not outline the method for doing this. At the conclusion of the paper, Dalley discusses some of the human factors connected with scheduling and states that these should be taken into consideration[9] yet his shift scheduling procedure does not embody these. An example of a schedule contained in Dalley's paper is contained in Figure 7.

2.2.2 Heuristic Procedures

Smith [46] extended the work of Monroe [40], as well as that of Frances [18] and Howell [25], by programming the procedures but utilizing the analytical capabilities of the human scheduler through the use of an interactive set of routines. Smith's method is composed of the following five steps:

- Step 1 Smith sums up the daily shift requirements, determined by an external allocation procedure, and

----

[9] Dalley closes with the following quote: "Your men [sic] should not be asked to work a shift that you yourself are not prepared to work."

Week #	Mon	Tue	Wed	Thu	Fri	Sat	Sun
1	D	D	D	D	D	*	*
2	D	*	D	D	D	D	*
3	D	D	*	D	D	D	D
4	*	A	A	A	A	A	*
5	A	A	*	*	A	A	A
6	A	A	*	A	A	A	*
7	*	*	A	A	A	A	A
8	*	*	A	A	A	A	*
9	N	N	*	*	N	N	N
10	*	*	N	N	N	N	N

Notation: * - Day off

N - Night Shift

D - Day Shift

A - Afternoon Shift

FIGURE 7 - Example of Dalley Schedule

divides by five. If the quotient is not an integer, the scheduler is prompted whether the number of personnel should be the next highest or lowest integer and which specific shifts are to receive the increase or decrease in allccation.

- Step 2 This step determines the frequency of each regular day off pair by solving the set of equations given by Monroe (see Figure 5). Rather than using Monroe's algolrithm, the following equations are solved.

$$V_{k-1,k} + V_{k,k+1} = Q_k, \text{ for } k=2,3, \ldots, 6$$
$$V_{71} + V_{12} = Q_1$$
$$V_{67} + V_{71} = Q_7$$

The solutions are:

$$V_{71} = 1/2[Q_7+Q_1-Q_2+Q_3-Q_4+Q_5-Q_6]$$
$$V_{67} = 1/2[Q_6+Q_7-Q_1+Q_2-Q_3+Q_4-Q_5], \text{ etc.}$$

where V_{ij} is the frequency of the day off pair consisting of days i and j and

Q_j is the number of employees off on day j.

If negative frequencies result, the scheduler must alter the allocation by selecting which changes are to be made from the alternatives generated by the procedure. The frequency of Saturday-Sunday weekends (V_{67}) is then compared with the minimum specified and if the constraints are not met, the scheduler must either accept the reduced frequency or modify the allocation,

possibly with the infusion of part-time staff, necessitating a return to Step 1. At this point, the first stage of shift scheduling, the selection of days-off periods, has been completed.

- Step 3 In this step, the rotation of days-off periods is generated by the following procedure for the case where regular days off must occur in pairs (ie. result of Step 2). The days-off pairs are selected sequentially[10], advancing one day for each week until a weekend (Saturday-Sunday pair) is required.[11] This weekend is inserted into the schedule and the days-off pair advances to Sunday-Monday. Whenever no occurrences of a particular days off pair remain, the days-off pair advances until an available pair is found. An example of a schedule completed thus far is given in Figure 8. One property of schedules produced by this method is that weekends, and occasionally long weekends (3 or 4 days in length), follow the longer working stretches. Smith states that this procedure also tends to reduce variations in the length of working periods. This does not appear to be the case in Figure 8. The lengths of working periods range from 2 to 10 days with the longest stretches (caused by going from a Monday-Tuesday or Tuesday-Wednesday to a Saturday-Sunday days-off period) all occurring in the beginning portion of the schedule.

---

[10] The days-off pairs are sequenced: Monday-Tuesday, ... , Saturday-Sunday, Sunday-Monday.

[11] Smith's scheduling problems included a "minimum frequency of weekends" constraint.

Week #	Mon	Tue	Wed	Thu	Fri	Sat	Sun
1	*	*					
2		*	*				
3						*	*
4	*						*
5	*	*					
6						*	*
7	*	*					
8		*	*				
9						*	*
10	*	*					
11		*	*				
12						*	*
13			*	*			
14				*	*		
15						*	*
16			*	*			
17				*	*		
18						*	*
19			*	*			
20				*	*		
21						*	*
22					*	*	
23						*	*

Legend: * - Day off

FIGURE 8 - Example of Smith Schedule

Also, although Saturdays and Sundays are spaced fairly evenly throughout the schedule, the other days of the week are not. In the example given, an employee would work fifteen weeks without a Thursday off while for Mondays, Tuesdays and Fridays the number of weeks would be 13, 12 and 14, respectively.

- <u>Step 4</u> Smith assigns the shifts to be worked to the working days by either of the following two procedures. If it is desirable for a single shift to be worked for as long as possible, the first shift in a day is assigned starting with the first day in the schedule, and continues until a day is encountered in which all of the first shift requirements have been included. Then the second shift commences to be assigned. This and the remaining shifts are handled similarly to the first shift, with the first shift being assigned once again in sequence after the last shift. This procedure will introduce, on occasion, occurrences where the amount of time between shifts is not sufficient[12]. These precedence violations of juxtaposed shifts are handled in the next step.

  If more frequent shift changes are desired, the shift advances to the next shift after the second day off every week in addition to the criteria given above.

- <u>Step 5</u> Due to a lack of suitable mechanical rules for

[12] If a 4 PM to Midnight shift on one day is followed by a Midnight to 8 AM shift on the next day, there will be no time off between these two shifts which will undoubtedly violate a working condition.

elimination of precedence violations, Smith's procedure identifies their occurrences and has the user specify "vertical interchanges" of shift assignments in other weeks in order to resolve them. This procedure is repeated until no precedence violaticns remain. In this step, the interactive program also prompts the human scheduler for interchages in order to make aesthetic improvements, such as reductions in the length of working periods.

If split or non-consecutive days-off pairs have to be used because of constraints on the lengths of working periods, Smith outlines two trial and error approaches whereby this may be accomodated. In determining the days-off pairs (Step 2), rather than specifying consecutive pairs, one can specify any seven pairs. Let the vectors V and Q be defined as before and define A as the 7 by 7 matrix where $A_{ij} = 1$ if day i is included in days-off pair j and $A_{ij} = 0$ otherwise. In order to find the frequency of the days-off pairs, $V = A^{-1}Q$, the matrix A obviously must be non-singular. If the set of pairs originally specified by the user result in a singular A matrix, the user is prompted for alternative pairs until a solution can be found.

The second approach is to determine the rotation of the days-off pairs (Step 3) with the constraints on the maximum lengths of working periods relaxed. Then, by trial and error, the user must interchange vertically, days off with days worked ensuring that no new violaticns of working

period length are introduced. As an example, the ten day stretch commencing in week 5 in Figure 8 could be broken up by interchanging weeks 5 and 22 for days 2 and 6. Weeks 5 and 22 would now be as shown in Figure 9.

Guha and Browne [19] describe some ad hcc procedures that produce schedules similar to those of Monroe. In order tc produce more feasible schedules, they include more than the regular days off by compensating for statutory holidays but this approach would be in conflict with some labour contracts with respect to paid holidays. They use an algorithm due to Tibrewala, Phillippe and Browne [47] to adjust the manpower allocation in such a way that days off can be scheduled consecutively.

Kregeloh and Miodrag [35] describe a heuristic procedure that generates a schedule satisfying the following constraints:

1. all working periods are six days in length
2. all day off periods are at least two days in length
3. all working periods commence on a weekday

This procedure handles several starting times during the day (vs. just three shifts), of which there are several in their examples because it was developed for transit operators. This method devises working periods that commence with shifts starting in the late evening and advances the starting times so that the working period finishes with an early morning shift.

Bennett and Potts [8] specify the shift to be worked as

Week #	Mon	Tue	Wed	Thu	Fri	Sat	Sun
1	*	*					
2		*	*				
3						*	*
4	*						*
5	*					*	
6						*	*
7	*	*					
8		*	*				
9						*	*
10	*	*					
11		*	*				
12						*	*
13			*	*			
14				*	*		
15						*	*
16			*	*			
17				*	*		
18						*	*
19			*	*			
20				*	*		
21						*	*
22		*			*		
23						*	*

Legend: * - Day off

FIGURE 9 - Modified Smith Schedule

an initial condition and then proceed to determine the days-off periods and their rotation throughout the schedule. Bennett and Potts treat each fortnight in a manner analagous to the way in which a week is treated in other papers; a PM shift is worked during the first seven days and an AM shift during the last seven. They formed the following objectives after discussions with the workers involved:

1. PM Sundays off should be spread evenly throughout the schedule.
2. (a) It should contain as many 4-day days-off periods as possible.

   (b) It should contain as many 3-day days-off periods as possible.

   (c) The 4-day periods should be spread evenly.

   (d) The 3-day periods should be placed so that the 3-day and 4-day periods are spread evenly.
3. Saturdays off should be placed so that weekend work is distributed evenly throughout the schedule, with some preference to Saturday-Sunday off together.
4. (a) It should contain as many 2-day days-off periods as possible.

   (b) The various days-off configurations should be distributed throughout the schedule.

They also had the constraint that there would be two days off in every Sunday to Saturday (PM or AM) week.

For the first stage of determining the days-off periods, they apply objectives 2(c), 2(d) and 4(a)

sequentially. Because in their particular application AM Sundays were always off, there is only one possible 4-day period and three 3-day periods, which are shown in Figure 10. Taking into account their specific application, Bennett and Potts give the following expressions for the maximum number of these days-off periods to occur, with 4-day periods having priority over 3-day ones to satisfy objectives 2(a) and (b)[13].

Let A_4, B_3, C_3, D_3 = the maximum number of days-off periods A,B,C,D respectively

y_i = number of times of day i off on PM shift, i=1,2,...,7 for Sunday, Monday,..., Saturday

z_i = number of times of day i off on AM shift.

$A_4 = \min(y_7, z_6)$, $B_3 = \min(y_6, y_7)$, $C_3 = 0$, and $D_3 = \min(y_1, y_6, z_7)$.

Before determining the remaining days-off periods, they insert the 3-day and 4-day periods by a procedure that ensures that the maximum will be formed. Objectives 1, 2(c) and 2(d) are satisfied by the use of spreading equations of the form:

Define a sequence $\{a_k\}$ by the relation

$a_k = [km/n] - [(k-1)m/n]$, k=1,2,...

where n is the number of weeks that m periods are spread over. Assign the period to week k iff $a_k=1$. [14] The PM Sundays (all AM Sundays are off) are assigned according to

---

[13] Bodin [10] presents an alternate strategy by sequentially solving three integer programs.

[14] The index k can be offset by a constant to allow a different starting point for the sequence.

Type	PM							AM						
	S	M	T	W	T	F	S	S	M	T	W	T	F	S
A	0	0	0	0	0	*	*	*	*	0	0	0	0	0
B	0	0	0	0	0	*	*	*						
C							*	*	*	0	0	0	0	0
D								*	0	0	0	0	0	*
	*	*	0	0	0	0	0	*						

Notation: * - Day off

0 - Day on

- Either On or Off

Note: AM Sundays always off

FIGURE 10 - Types of Long Weekends

the above sequence, then the 3-day and 4-day periods are "built" around them. After this, the remaining Saturdays are assigned by first ensuring that no more than two weekend days are worked in a fortnight, then forming as many Saturday-Sunday pairs as possible, then attempting to reduce those fortnights with two weekend days worked to only having one worked and then, if possible, none worked.

The remaining days-off periods are determined by an integer program that maximizes the number of 2-day periods. These are then spread throughout the schedule using a spreading sequence as described above.

Bennett and Potts then describe a procedure to assign actual transit routes, along with starting times, to the positions in the schedule. They formulate a multi-index assignment problem with the objective being to minimize overtime and maximize the weighted sum of two measures that reflect the number of pairs of days in the same week with the same shift and the number of pairs of days in the same week with a similar starting time. Because this problem is too large to feasibly solve, they describe a simpler procedure that does not guarantee optimality but with the examples that they used, they found solutions with zero overtime. They state that because the other measures are only approximate evaluations of the workers' preferences, optimization is not warranted.

Bodin in [10] describes a method developed in an earlier paper [11] for use by the New York City Department

of Sanitation. In determining the days-off periods, Bodin adopted the following strategy. Since Sundays were always off, Bodin considered single days off (days-off periods 1 to 7 for Monday to Sunday, respectively) and the following days-off clusters: Friday-Saturday-Sunday, Saturday-Sunday, Sunday-Monday and Sunday-Monday-Tuesday (numbered 8 to 11, respectively). These latter four clusters were assigned values V_8 to V_{11} to reflect their relative benefit, although Bodin gives no method of determining these values. If $V_8 > V_9$ as many Fr-Sa-Su periods were formed as possible and if $V_8 < V_9$, then Sa-Su clusters were formed. Similarly, if $V_{10} > V_{11}$, Su-Mo clusters were formed and if $V_{10} < V_{11}$, Su-Mo-Tu periods were formed. In [10], Bodin develops a general approach for this first stage in the form of the following integer program.[15]

$$\text{Maximize} \quad \sum_{t=1}^{11} V_t X_t$$

$$\text{Subject to} \quad \sum_{t=1}^{11} A_t X_t = r_j, \; j=1,\ldots,7$$

$$\underline{X}_t \le X_t \le \overline{X}_t, \; X_t \text{ integer}$$

Where X_t is the frequency of the t'th days-off period
r_j is the requirement for day j

Bodin shows that the procedure given above yields the same solution as the integer program with the values of the single days off (V_1 to V_7) set to zero.

---

[15] Conditions are given in Chapter 3 for which the integer solution is guaranteed by solving it as a linear program.

For the second stage of developing the rotation of the days-off periods, Bodin uses the following spreading equation to evenly distribute the K clusters longer than one day that resulted from the first stage. The kth cluster is assigned to week a_k where

$$a_k = (k-1)N/K \quad , \; k=1,2,\ldots,K$$

where N is the number of weeks.

If K=8 and N=30, this would yield $a_k=\{1,4,8,12,16,19,23,27\}$. In an example given in [10], there were 8 clusters consisting of 6 Saturday-Sundays and 2 Sunday-Mondays. These latter two were assigned to weeks 1 and 4 (there were 30 weeks in the schedule) and the Saturday-Sundays assigned to the remainder. Since all Sundays are off for this application, the remainder of this stage consists of assigning the single days off of Tuesday to Friday.[16] Bodin accomplishes this by first assigning all Tuesdays to weeks with only one day, starting at the beginning of the schedule. This is then followed by the Wednesdays, Thursdays and Fridays. Bodin was not concerned with the assignment of shifts so there was no third stage. An example of a completed schedule is shown in Figure 11.

In [10] Bodin describes a somewhat general formulation of the second stage, the generation of the days-off periods, which is essentially due to Heller [22]. This formulation is discussed in the Enumerative Procedures section below along with the remainder of Heller's procedure. A formulation more

[16] There are no single Mondays or Saturdays because they are always adjacent to a Sunday.

Week #	Mon	Tue	Wed	Thu	Fri	Sat	Sun
1	*						*
2		*					*
3		*					*
4	*						*
5		*					*
6			*				*
7			*				*
8						*	*
9			*				*
10			*				*
11			*				*
12						*	*
13				*			*
14				*			*
15				*			*
16						*	*
17				*			*
18				*			*
19						*	*
20				*			*
21				*			*
22					*		*
23						*	*
24					*		*
25					*		*
26					*		*
27						*	*
28					*		*
29					*		*
30					*		*

Notation: * - Day off

FIGURE 11 - Example of Bodin Schedule

general than that of Heller is described in Chapter 3 of this thesis.

2.2.3 Enumerative Procedures

The procedures developed by Heller along with McEwen and Stenzel [22, 23, 24] address both the manpower allocation and shift scheduling problems. These have been used by the St. Louis Police Department, among others. Only the shift scheduling procedure will be described in this thesis. This procedure is compared in Chapter 5 with the new method described in Chapter 3.

In order to make use of Heller's shift scheduling procedure, the manpower allocation must be able to be split up so that a whole number of employees can be assigned to each shift for the entire week. Heller then applies these allocations to the shifts and determines the days-off periods and their rotation for each shift individually. For each shift, the scheduler specifies maximum and minimum values for the number of days off at the beginning and end of the shift, called b and e, respectively. As each shift occupies an integral number of weeks of the schedule, each (b,e) pair specifies particular days of the week. Thus (1,2) would refer to Monday off at the beginning and Saturday-Sunday off at the end.[17] For each feasible pair (b,e), the method determines all the ways in which the

---

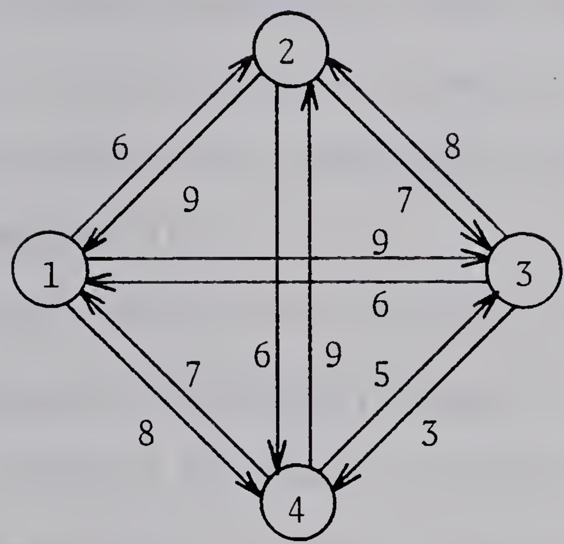
[17] A week, as used by Heller, commences on a Monday and ends on a Sunday.

number of remaining days off can be partitioned into a set of numbers, each of which is an acceptable length of days-off period. For example, if two is the minimum length of a days-off period, four days can be partitioned into two groups of two days or one group of four. For each partition, it determines all the ways the days off pericds of appropriate lengths can be formed from the available days off. For example, if the partition is one period of length three and one of length two and the available days off are Wednesday through Sunday, the days-off periods can either be Wed-Thu-Fri and Sat-Sun or Wed-Thu and Fri-Sat-Sun. Each collection of days-off periods can be represented as a cyclic graph where the nodes represent each days-off period and there is a directed arc from one node to another if a working period of acceptable length can be formed between that days-off period and the other. The weights of the arcs are the lengths of these working periods.[18] An example from Heller [23] is shown in Figure 12. The problem now for each graph, is to find all possible Hamiltonian cycles[19] such that the sum of the weights equals the total number of working days for this shift. Heller finds all sequences of days-off periods by representing the graph as a separation matrix, D, where D_{ij} = infinity if no arc exists from node i to node j and D_{ij} equals the weight of the arc if one does

---

[18] There is an arc from the last days-off period, corresponding to the e days off at the end, to the first days-off period, corresponding to the b days off at the beginning. The weight of this path is zero.
[19] A Hamiltonian cycle is a cycle where all nodes in a graph are visited once and only once.

$$D = \begin{pmatrix} \infty & 6 & 9 & 8 \\ 9 & \infty & 7 & 6 \\ 6 & 8 & \infty & 3 \\ 7 & 9 & 5 & \infty \end{pmatrix}$$



If sum of weights must equal 30 (i.e., the total number of working days), two possible paths are:

1-3-2-4 and 1-4-3-2

NOTE: This example is based on a rotation for only one shift. Thus there are no nodes corresponding to b and e. (i.e., in the above two paths, node 1 follows the last node.)

FIGURE 12 - Heller's Separation Matrix and Graph

exist. The problem is now to select a set of n non-infinite entries of the n x n matrix D[20] such that only one entry in each row and each column is selected and the sum of the entries chcsen equals the total number of working days. Heller determines these sequences by complete enumeration. If all entries were non-infinite, there would be at most n! sequences. However, this number is considerably reduced because there is at least one infinite entry in each column because D_{ii} = infinity.[21]

Heller ranks the sequences produced using a lexicographic objective function, shown in Figure 13, containing such measures as number, length and type of weekend days-off periods[22], number of consecutive working weekends and lengths of working periods. The best schedules for each shift are ccmbined to form complete schedules by enumerating those combinations where the length of the days-off period at each shift change is acceptable. These complete schedules are then ranked by another lexicographic objective function similar to the one above.

The lexicographic objective function has theoretical shortcomings. Because of the inter-relationships between the various measures, they cannot be viewed as independent. Using the lexicographic function, one schedule would be

---

[20] n is the number of nodes, ie. the number of days-off periods.

[21] This is because there cannot be a path from a days-off period to itself. If there is more than one occurrence of a particular set of days off, these are represented as different periods and thus different nodes on the graph.

[22] Type as in whether it would be a Fri-Sat-Sun, a Sat-Sun-Mon, etc.

Factors in order of decreasing significance:

-Number of weekend days-off periods (DOP's)

-Maximum number of consecutive working weekends

-Maximum length of working period (WP)

-Frequency of each of the following DOP's:

(each frequency is a factor)

- Fri-Sat-Sun-Mon
- Thu-Fri-Sat-Sun
- Sat-Sun-Mon-Tue
- Fri-Sat-Sun
- Sat-Sun-Mon
- Sat-Sun
- Wed-Thu-Fri-Sat
- Thu-Fri-Sat
- Sun-Mon-Tue-Wed
- Sun-Mon-Tue
- Fri-Sat
- Sun-Mon

-Range of working period length

-Maximum sum of lengths of two consecutive WP's

-Frequency of above maximum sum

-Standard deviation of (WP length/following DOP length) ratio

FIGURE 13 - Lexicographic Objective Function

rated better than another if the first one had a slightly higher value for the most significant measure, even though the second schedule could have substantially higher values for all the other measures. An illustrative example can be created using as measures the frequencies of twelve different types of weekends under consideration. Using the preference ordering of the weekends as used by Heller (see Figure 13), under the lexicographic function, a schedule with one Fri-Sat-Sun-Mon weekend would be ranked higher than a schedule with four weekends of Thu-Fri-Sat-Sun but no Fri-Sat-Sun-Mon weekend, regardless of the rest of the schedules[23]. It is unlikely that this ranking would reflect the preferences of a group of workers.

One restriction imposed by the separation matrix representation of the cyclic graph, used to determine the rotation of the days-off periods, is that for any two periods, there must be at most one intervening working period of acceptable length. This is equivalent to there being at most one arc connecting any two nodes on the graph. An example where this condition is not satisfied is a case where working periods of length two days and nine days are acceptable. In this case there is not one unique value for the separation between periods such as a Tue-Wed off and a Sat-Sun off.

All of the schedules that are produced by the above

---

[23] This is assuming that the four measures of higher significance were equal. One can easily construct an example similar to the one above where these measures are equal.

method have similar characteristics. There is only one period, consisting of an integral number of weeks, on each shift during a rotation. If the length of the rotation, ie. the number of groups of employees, is large, there could be an undesirable length of time between working periods of a particular shift. For example, if there was a rotation of eighteen weeks equally divided between three shifts of nights, days and afternoons, there would be stretches of twelve weeks without a day shift. Heller acknowledges this and suggests that each shift can be subdivided into two or more "shifts" and use the entire method treating each of these "shifts" separately. However, Heller suggests only a trial and error approach to the splitting up of the shifts. Even a small case with three shifts, each three weeks in length, can have up to 125 different subdivisions, each of which would represent a complete problem for his method.[24]

Assigning complete weeks to the shifts does not guarantee the maximum number of weekends, which is the most significant measure in Heller's objective function. Unmatched Saturdays and Sundays off can result if more Saturdays than Sundays off are assigned to one shift while more Sundays than Saturdays off are allotted to another. Another result of the shift assignment by weeks is that all shift changes occur over the end of the week (Sunday to

----

[24] Each shift of 3 weeks can be partitioned in five ways: 123, 12 & 3, 1 & 23, 13 &2, and 1 & 2 & 3 where, for example, 1 & 23 means the three week period is split after week 1. Thus the total number of combinations for the three shifts is 5**3 = 125.

Monday); no schedules with shift changes in the middle of the week will be generated.

2.3 Methods for Individual Schedules

2.3.4 Trial and Error Procedures

Morrish and O'Connor [41] also develop schedules in which the days off rotate but the shifts are individually assigned. The conditions under which the employees, in their application nurses, are scheduled are:

1. a weekend off every three weeks
2. no more than six working days in a row
3. nurses either alternate between day shift and another shift (either nights or afternoons but not both) or can be permanently assigned to a non-day shift.

The method for generating the days off rotations was not specified but the shift assignments were made by priority of the unit (ie. how critically a nursing station needed the personnel) and by category (ie. Registered Nurses, Practical Nurses and Licensed Nursing Aides). The frequency of periods on nights or afternoons for each worker was taken into account in an attempt to be equitable to all staff members. Also nights and afternoons were assigned first so that if there was to be a shortage, it would occur on day shift when

there is more personnel scheduled.[25]

Ahuja and Sheppard [3] discuss the generation of schedules and calendars for individual employees as well as the comparison of actual allocation of staff with desired. They do not state how the days-off periods and rotations are formed and the programs require much interaction from the user.

2.3.5 Heuristic Procedures

Baker and Magazine [7] are primarily concerned with developing lower bounds on the size of the workforce necessary when the daily requirements for Monday to Friday are at one level (N) and at another (n) for Saturday and Sunday, where N and n can be equal. They discuss four different sets of days-off conditionsd give an algorithms that generate one schedule which achieves the lower bound and satisfies the conditions. The four constraints are:

1. two days off each week
2. two consecutive days off each week
3. every other weekend off and four days off every two weeks
4. every other weekend off and two pairs of consecutive days off every two weeks.

The schedules are not rotational and an employee is expected

---

[25] Obviously this procedure could not be used for cases where there were more personnel required on a shift other than day shift.

to repeat the days off every week for cases (1) and (2) and every two weeks for cases (3) and (4). As an example, the algorithm for the first constraint is given below:

Let W be the size of the workforce.

1. Assign the first (W-N) employees in the schedule Monday off.
2. For the remaining weekdays, assign the next (W-N) employees the next day off, where employee 1 is next after employee W.
3. For the remaining (weekend) days, assign the next (W-n) employees Saturday off and the following (W-n) employees Sunday off.
4. If 5W > 5N+2n, there will remain some additional workdays to be assigned. These can be filled in arbitrarily so that there are five working days for each employee.

An example of a schedule generated by this method is shown in Figure 14. Baker and Magazine admit that it is optimal only with respect to workforce size and that its primary virtue is simplicity. They do not discuss the assignment of shifts.

Luce [36] describes a heuristic procedure that was used to schedule telephone operators (see Buffa et al. [12]). He attempts to maximize the number of consecutive days-off pairs while ensuring that at least one day is off each week. For cases where consecutive days off are not possible, he maximizes the number of working days between days off. Days

Employee #	Mon	Tue	Wed	Thu	Fri	Sat	Sun
1	*			*			
2	*				*		
3		*			*		
4		*				*	
5			*			*	
6			*				*
7				*			*

Notation: * - Day off

Note: This is an individual schedule. For example, employee #1 always has Mondays and Thursdays off for the time that this schedule is in effect.

FIGURE 14 - Example of Baker and Magazine Schedule

off are rotational so that all employees benefit from occasional weekends off but the shift assignments are made to the highest ranked shift available for each operator on a seniority basis. Workers' preferences are measured by an ordinal ranking of each shift for each day of the week. As implemented in [12], lunch and coffee breaks are also specified for each operator.

2.3.6 Enumerative Procedures

Miller [39] has developed a mathematical programming formulation for developing shift schedules on an individual basis over a short scheduling horizon. He first separates all constraints and preferences into two classes: binding and non-binding constraints. For each employee, he creates a set π_i of all patterns of days on and off for the horizon that satisfy the binding constraints. He does not state how this set is generated but gives an example with a fourteen day horizon that has a maximum of 1001 possiblities of days on and off over the period. Taking into account the constraints and special requests[26], this number is reduced considerably. The set of non-binding constraints consists of two groups: staffing level constraints and pattern constraints. By assigning manpower staffing and shortage costs and costs of violating pattern constraints, Miller develops a penalty function, shown in Figure 15. This

[26] An employee is able to incorporate special days off directly into the schedule.

The penalty function formulation is as follows:

Find $x^1, x^2, \ldots, x^I$ which maximize

$$\lambda \left\{ \sum_{k=1}^{14} f_k \left(\sum_{i=1}^{I} x^i\right) + \sum_{k=1}^{14} \sum_{j \varepsilon J} h_{jk} \left(\sum_{i \varepsilon B_j} x_k^i\right) \right\} +$$

$$(1-\lambda) \left\{ \sum_{i=1}^{I} A_i \cdot \sum_{n \varepsilon N_i} \alpha_{in} \cdot g_{in}(x^i) \right\}$$

subject to $x^i \ \varepsilon \ \Pi_i$, $i=1, \ldots, I$

where $x^i = (x_1^i, \ldots, x_{14}^i)$ where x_k^i = 1 if day k is worked by employee i
= 0 otherwise

$f_k\left(\sum_{i=1}^{I} x_k^i\right)$ is the staffing cost of having $\sum_{i=1}^{I} x_k^i$ employees on day k

$h_{jk}\left(\sum_{i \varepsilon B_j} x_k^i\right)$ is the staffing cost of the j'th subgroup of employees (eg. RN's, RNA's, etc.) of having $\left(\sum_{i \varepsilon B_j} x_k^i\right)$ members of that subgroup working on day k. (B_j is the set of employees of subgroup j).

$g_{in}(x^i)$ is the cost of violating the n'th pattern constraint.

α_{in} is the measure of importance of the n'th constraint for the i'th employee.

N_i is the set of pattern constraints of concern to employee i.

A_i is a measure of how good the past schedules have been vis-a-vis the preferences of employee i.

λ $0 \leq \lambda \leq 1$ gives a relative weight of the staffing level costs versus the pattern costs.

FIGURE 15 - Miller's Objective Function

function also takes into account each employee's preferences and the "goodness" of each employee's past schedules. However, he does not specify how one is to calculate these costs and the appropriate weights for each employee.

Miller solves the above problem by a cyclic coordinate descent algorithm that attempts to find a better schedule from π_i for each of the i of {1,...,I} employees in sequence and terminates when no improvement is made after I iterations. This procedure must be supplied with an initial feasible solution. Miller mentions that more than one shift can be accomodated by adjusting the set of feasible schedules for each employee (π_i) but does not state how this could be accomplished. However, he has developed an enumerative procedure that has allowances for individual's preferences. If the number of employees involved is small and the planning horizon short, the problem is feasible to solve. Of course, the validity of the solutions is heavily dependent on the values given to the weights and costs.

2.4 Procedures Concerned Only With Allocation

As stated earlier, the shift scheduling problem can be defined as the generation of shift schedules that satisfy an allocation of employees to the various shifts, in addition to conditions imposed by the employees. Although the main thrust of this thesis is with the generation of shift schedules, the following allocation procedures are mentioned because they do have an impact on the type of shift schedule that ultimately might be produced. Much of the research into the allocation of manpower has been very specific to the application in order to take advantages of each situation. Abernathy et al. [1], Harris [20], McCartney et al. [38] and Warner and Prawda [48] dealt with the allocation of nursing personnel and were concerned with the forecasting of requirements and allocation of several classes of nurses. These classes, such as Registered Nurses and Registered Nursing Aides, could not be considered separately because a shortage in one class could be partially made up by a surplus in another. Chaiken et al. [15] in the Patrol Car Allocation Model use a queueing model to determine the number of patrol units needed in order to achieve a particular level of response while Church [14] uses time series analysis to forecast the requirements of telephone personnel.

However, because the manpower allocation and shift scheduling problems are not entirely independent, one can not use the solutions of the above-mentioned procedures and

simply apply any shift scheduling procedure. The method of Warner and Prawda was designed to be used twice a week which precludes the use of rotational schedules. With the other procedures, there is no guarantee that a feasible schedule exists that satisfies all the working condition constraints and the allocation. Tibrewala et al. [47] developed an algorithm which for a given allocation will produce a revised allocation that minimizes the total personnel required while meeting the original allocation and ensuring that at least one shift schedule can be devised that will guarantee two consecutive days off each week. Baker [5] also is concerned with the condition of consecutive days off and discusses an allocation procedure that accomplishes this that also allows the inclusion of part-time employees. A general overview of the manpower allocation problem along with a brief discussion of the shift scheduling problem is given in Baker [6].

2.5 Summary of the Literature Review

Much of the shift scheduling literature describes methods that were developed for specific scheduling situations. These methods are restrictive, both in terms of the scheduling problem and the types of schedules generated, in order that realistic problems could be solved within a reasonable length of time. These restrictions preclude the general applicability of the methods. Those that did not impose many restrictions were essentially trial and error procedures that relied on the human scheduler. These methods have drawbacks as general methods due to human limitations; as well, they cannot be automated because they are not proper algorithms. Only two methods consisted of algorithms which enumerated schedules. One of these, developed by Heller et al. [22, 23, 24] is restricitve in the types of schedules produced; the other, by Miller [39], is more general but generates individual schedules.

Bodin [10] addressed the problem of developing a general shift scheduling method when he devised a general model in which he imbedded four existing methods. The work of Bodin is extended in this thesis. A general method is developed for generating cyclic schedules; this methcd is described in the next chapter.

The shift scheduling problem was described in Chapter 1 as being composed of three stages. The third stage, which consists of assigning the shifts to wcrking days, has rarely been dealt with in the literature. Heller's method restricts

the shift assignment to a specific form prior to the use of the method; Miller's method was developed for single shift situations. Other authors, when they mention the shift assignment problem at all, suggest arbitrary and trial and error procedures. In response to this, most of the development of the new method centred on the third stage.

Another issue ignored by many authors is the incorporation of workers' preferences. Some, such as Smith [46], suggest a trial and error approach to adjust the schedule produced while others, such as Baker and Magazine [7] impose a small number of constraints reflecting the wishes of the workers. Bennett and Potts [8] incorporate certain preferences of the employees into an objective function. The method of Miller and the Three Stage Method described in the next chapter allow general utility functions in addition to constraints; however, procedures for generating these utility functions have not yet been developed. This latter subject is discussed in the final chapter as an area of future research.

CHAPTER 3 A Three Stage Method for Generating Shift Schedules

3.1 Overview of Method

In this chapter, a new method for producing shift schedules is described. Using the categories defined in the last chapter, this method would be classified as a cyclical, enumerative procedure.

As described in Chapter 1, the Three Stage Method consists of the following stages:

Stage 1

The days that are to be off during the schedule duration, obtained from the manpower allocation, are grouped into days-off periods.

Stage 2

The days-off periods from Stage 1 are ordered to form rotations by specifying sequences of days-off periods and working periods in the schedule.

Stage 3

For each working day in the rotations generated by Stage 2, the shift to be worked by a worker is specified. This stage is not necessary if there is only one shift worked.

Stage 1 is formulated as an integer linear program. Stage 2 and 3 generated alternatives by tree based enumeration procedures. Examples of the cutcomes of each stage are shown in Figures 16 to 18.

3.2 Stage 1: Determining the Days-off Periods

The objective of this stage is to group the days off into days-off periods such that the total value of these periods is maximized according to some objective function. To determine such a set of days-off periods for the whole schedule, two pieces of information are needed. Firstly, over all weeks in the schedule the frequency that each day of the week (eg. Monday) is specified to be a day off; and secondly, the worth or value of each potential DOP. The first part is determined from the manpower allocation (see Figure 16) while the second part involves measuring the relative utility or benefit of each potential DOP to the entire group of employees. Ideally, a thorough multi-attribute utility analysis should be performed to determine these values. For applications in this thesis, utility values (ie. numbers) were chosen to reflect employees' preferences over a small number of attributes, such as the length of a days-off period and whether the days-off pericd contains a weekend. For example, if four-day periods are preferred to two-day periods then the value of a four-day period would be more than twice the value of a two-day period.

For the following manpower allocation: 5 groups of employees to work two shifts and have the specified frequency of specific days off.

	Mon	Tue	Wed	Thu	Fri	Sat	Sun
Days - D	2	2	2	2	2	2	2
Afternoons - A	1	1	2	2	2	1	0
Day off	2	2	1	1	1	2	3

One set of days-off periods satisfying this manpower allocation is shown below:

#	Days-off period
1	Fri-Sat-Sun
2	Mon-Tue-Wed
3	Sat-Sun
4	Sun
5	Thu
6	Mon-Tue

FIGURE 16 - STAGE 1 - Determination of Days-off Periods

One rotation of the days-off periods given in Figure 16 is:

#	days-off period	length of preceding working period
6	Mon-Tue	4 days
4	Sun	4 days
3	Sat-Sun	5 days
1	Fri-Sat-Sun	4 days
5	Thu	3 days
2	Mon-Tue-Wed	3 days

This may be shown as follows:

Week #	Mon	Tue	Wed	Thu	Fri	Sat	Sun
1	*	*					*
2						*	*
3					*	*	*
4				*			
5	*	*	*				

Legend: * - Day off

NOTE: The 4-day working period preceding the Mon-Tue days-off period in week #1 is Thu to Sun in week #5 as the schedule is cyclic. The index of a days-off period is also the index cf the preceding working period. Thus WP # 1 is Thur-Sun in week # 5.

FIGURE 17 - STAGE 2 - Generation of Rotations

One feasible shift schedule for the rotation given in Figure 17 that satisfies the manpower allocation given in Figure 16 is given below:

Week #	Mon	Tue	Wed	Thu	Fri	Sat	Sun
1	*	*	A	A	A	A	*
2	A	A	A	A	A	*	*
3	D	D	D	D	*	*	*
4	D	D	D	*	D	D	D
5	*	*	*	D	D	D	D

Legend: * - Day off

D - Day shift

A - Afternoon shift

Note: This schedule contains the minimum number of shift changes in the cycle.

FIGURE 18 - STAGE 3 - Assignment of Shifts

The above problem may be formulated as the following integer linear program (ILP), based on the one developed by Bodin [10]. If the restriction is imposed that days-off periods are not to be longer than the number of days in a week (as Bodin implicitly assumes), then a matrix A can be defined as follows.

Given T potential, or acceptable, DOP's, each element

A_{it} = 1 if the t'th potential DOP contains day i of the week for $t=1,2,\ldots,T$

= 0 otherwise.

In addition, define the vectors V, R and X where

V_t is the value or utility of the t'th potential DOP

R_i is the frequency of day i being a day off in the manpower allocation.

X_t is the frequency that the t'th potential DOP is included in the collection.

Thus the ILP is

$$\text{Maximize} \quad \sum_t V_t \bullet X_t \tag{1}$$

subject to

$$\sum_t A_{it} \bullet X_t = R_i \qquad \text{for } i=1,2,\ldots,7 \tag{2}$$

$$X_t \geq 0 \text{ and integer} \tag{3}$$

If lower and upper bound constraints are specified for each X_t, the above ILP is identical to that of Bodin. However, the formulation is not complete. In the general case, there will be upper and lower bounds on the numbers of <u>working periods</u> of a particular length. From these bounds, an upper and lower bound on the total number of working periods can

be calculated. Clearly, the number of days-off periods in a cyclic schedule is the same as the number of working periods and therefore these upper and lower bounds have to be incorporated into the ILP formulation. As an example, let there be a constraint that for a schedule with 45 working days, the longest and shortest working periods allowed are 7 days and 4 days in length respectively. In addition, let there be no restriction on the numbers of working periods of length 4 to 7 days inclusive. These conditions imply a lower bound of zero on all lengths and an upper bound of zero on lengths outside the range 4 to 7. As there are 45 working days, there must be at least $\lceil 45/7 \rceil = 7$ and at most $\lfloor 45/4 \rfloor = 11$ working periods. Therefore the upper and lower bounds on the total number of days-off periods are 11 and 7, respectively. Let U and L represent the upper and lower bounds, respectively. Thus, the two additional constraints are

$$\sum_t X_t \geq L \tag{4}$$

$$\sum_t X_t \leq U \tag{5}$$

Although the above formulation (equations (1) to (5)) is an <u>integer</u> linear program, the ordinary linear programming solutions for all examples performed consisted only of integer values. It is suggested that this result might hold for all cases. Even if an integer linear programming method has to be used, this problem is small with only 9 constraints and as many variables as potential days-off periods.

If days-off periods longer than the number of days in a week are to be considered then the above formulation can be used with the A matrix redefined as:

A_{it} is the number of occurrences of day i in the t'th potential DOP.

It is suggested that the linear programming solutions to this problem will not necessarily consist of integer values.

Regardless of the procedure used to solve the formulation in equations (1) to (5), the result of this stage is the frequency of each days-off period to be included in the schedule. This is then used as input to Stage 2 to determine the rotation of the days-off periods. Only those DOP's with non-zero frequency need be considered in the next stage. The "number of unique DOP's" is the number of such DOP's, ie. T - the number with zero frequency. The "total number of DOP's" refers to the total frequency of DOP's, ie. $\sum_t X_t$.

3.3 Stage 2: Generating the Rotation of the Days-off Periods

The objective of this stage is to generate rotations containing the days-off periods produced by the first stage. Furthermore, these days-off periods must be sequenced so that the sum of the lengths of the intervening working periods is equal to the total number of working days specified in the manpower allocation; ie. the number of weeks in the rotation is equal to the number of groups of workers. The working periods must be of acceptable length, with an acceptable frequency of WP's of each length. In addition, constraints regarding the ordering of the working periods, such as no two consecutive WP's of length ≥ 7, or the days-off periods, such as weekends off not being consecutive, must be satisfied. The method essentially uses exhaustive enumeration but in a manner such that constraints may be easily incorporated in order to prune the enumeration process.

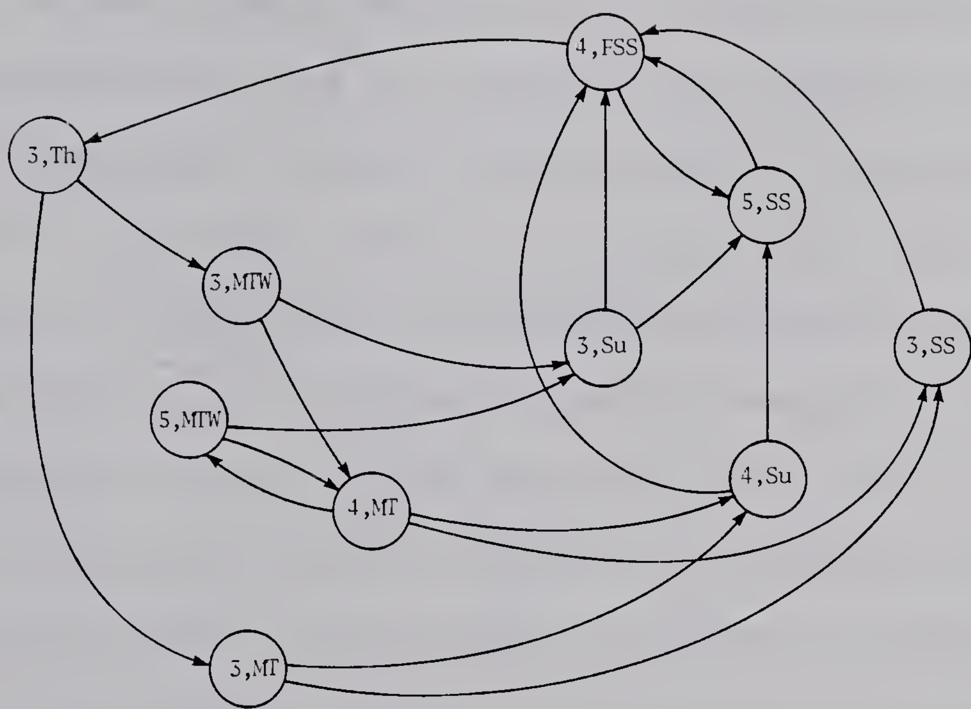
As stated earlier, a rotation can be uniquely represented by a sequence of ordered pairs (w,k) where w is the length of the WP preceding a specific DOP and k is the index of that DOP. These k's are from the set {1,2,...,K} where K is the <u>total</u> number of DOP's. As this represents a rotation, the first pair follows the last pair in sequence. This representation was chosen because it would easily accomodate constraints regarding a DOP and the preceding WP. An example of such a constraint is that eight days on must

be followed by at least three days off. Note also that for DOP's with frequency greater than one, each occurrence is indexed by a different k value.

Taking all acceptable pairs, a directed graph, G, can be constructed where there is an arc from (w_1 ,k_1) to (w_2, k_2) if $k_1 \neq k_2$ and w_2 is an allowable and possible length of a WP between DOP's k_1 and k_2 . There can be more than one possibility for w_2 . If WP's of both 2 and 9 nine days are acceptable, then between DOP's such as Monday-Tuesday and Friday-Saturday, there are two possible lengths and thus two arcs. An arc is not constructed if DOP k_2 may not directly follow DOP k_1 or a WP of length w_2 may not directly follow a WP of length w_1 . As cycles are to be found, any node (ie. pair) that does not contain at least one arc leading to it and at least one arc leading from it can be removed from the graph. The graph G for the example given in Figures 16 to 18 is shown in Figure 19.

The problem can now be stated as finding all cycles in G such that each DOP (ie. the k value) is to be included once and only once and the sum of the w values is to equal the total number of working days. This first part can be expressed as the length of the cycle is K arcs and for any two pairs on the path, (w_1 ,k_1) and (w_2 ,k_2), $k_1 \neq k_2$. In addition, constraints regarding the sequence of WP's of a particular length must be satisfied.

In order to reduce the size of the problem, a more compact but equivalent representation can be used. The



GRAPH G

Notation: node (w,k): w - length of preceding working period
k - days off period

Th - Thursday
MTW - Monday - Tuesday - Wednesday
MT - Monday - Tuesday
Su - Sunday
SS - Saturday - Sunday
FSS - Friday - Saturday - Sunday

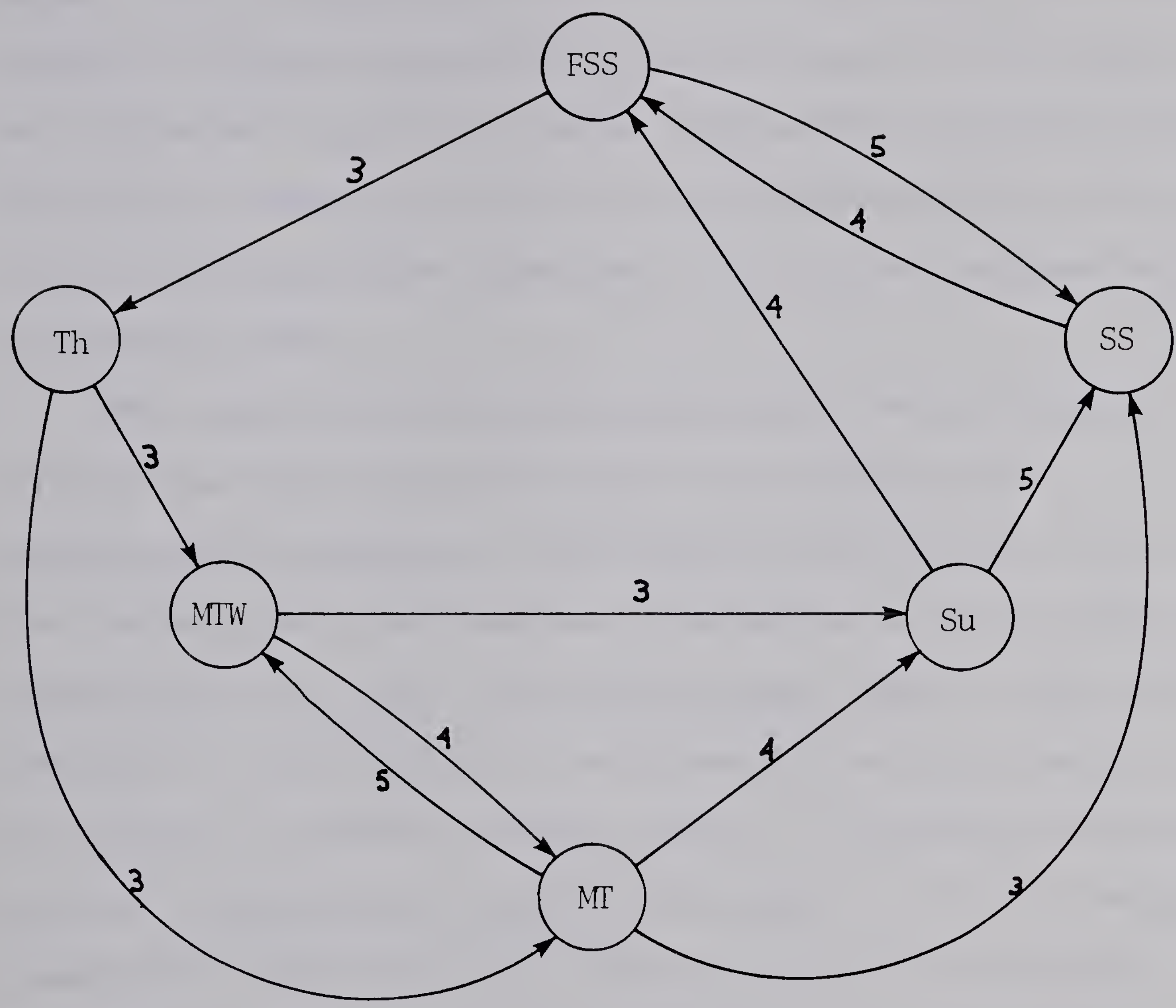
FIGURE 19

weighted directed graph G' is constructed as follows. For every node (w_1 ,k_1) in G, the node k_1 is constructed in G', if it hasn't already been created. If there is an arc from a node (w_1 ,k_1) to a node (w_2 ,k_2) in G, then an arc is constructed in G' from k_1 to k_2 with weight w_2 , if such an arc doesn't already exist. The graph G' shown in Figure 20 is equivalent to G in Figure 19. A Hamiltonian path[28] is a cycle that passes through each node in a graph once and only once. The problem can now be expressed as finding all Hamiltonian paths on G' such that the sum of the weights on a path equals the total number of working days. With this representation, constraints concerning successive lengths of working periods must be handled directly because they cannot be incorporated into G'. As an example, if no two consecutive WP's are to be of length ≥ 7, then at each point when selecting an arc, an arc with a weight ≥ 7 cannot be selected if the weight of the previously selected arc had a weight ≥ 7.

The general Hamiltonian cycle problem falls into the class of problems, called NP, for which it is conjectured that no known solution procedure exists that can be executed in an amount of time bounded by a polynomial of the problem size. The Stage 2 problem also belongs to this class because any Hamiltonian cycle problem can be reduced to this problem in the following manner: Assign weights to the arcs such that the sums of the weights on each Hamiltonian cycle are

---

[28]For elaboration see Aho et al. [2].



GRAPH G^1

Notation: node (k) - days-off period

weight on arc - length of working period between days-off periods

FIGURE 20

equal. (It is always possible to construct such a set of weights.) Place a constraint that the sum of the weights on any cycle must equal the above amount. This problem is now the Stage 2 problem. This is the justification for using the following inefficient algorithm, ie. one not polynomially bounded in time.

The basic algorithm for generating the Hamiltonian cycles, and thus the rotations, is an exhaustive enumeration. Starting at some node k (ie. DOP with index k), the enumeration tree fans out, including each node that is connected to the most recently included node. A node isn't included if the length of the preceding working period, ie. the weight on the arc, would violate a frequency constraint, such as a maximum of two WP's of length 7, or an ordering constraint, such as no two consecutive WP's of length $\geq$ 7. Any cycle generated with the sum of the weights of the arcs not equal to the total number of working days is discarded. However this procedure does not guarantee unique rotations. Duplicate rotations can be avoided by adopting the following approach: A node k is included only if all nodes k'<k which have the same days off as DOP k are already included.

The above procedure is similar to the one used by Heller [22]. Because his method imposes the restriction that there can be at most one possible length of WP between a pair of DOP's, a "separation matrix" can be created where D_{ij} is the number of days between DOP i and DOP j. If i = j, or if there is not an allowable length of WP between DOP's i

and j, D_{ij} is set to infinity. Heller states the problem as selecting one element from each row and each column so that the sum of the elements equals the total number of working days. The method used is an exhaustive enumeration.

The result of this stage is a set of rotations, ie. a set of alternating sequences of days-off periods and working periods. If there is only one shift, then these represent complete schedules. Otherwise, these rotations must be input to Stage 3 to determine the shift assignments.

3.4 Stage 3: Assignment of Shifts Worked

3.4.1 Overview of Algorithms

The purpose of this chapter is to specify which shifts are to be worked on the working days of the schedules. Given are rotations of working periods and days-off periods, such as those produced by Stage 2. The manpower allocation specifies the number of groups of employees working on each shift of each day of the week. For this section, a group is assumed to consist of one employee; this simplifies the wording without loss of generality. A shift must be assigned to a particular day of the week in the schedule as many times as there are people working it. The algorithms developed in this section generate several schedules for each rotation that satisfy the manpower allocation.

The following notation will be used in this section. From the manpower allocation, n_{ji} is the number of people that are specified to work shift j on day i. (Shift j = 1,2,...,J where J is the number of shifts and day i = 1,2,...,7 since there are 7 days per week.) Because the schedules are rotational, there are n_{ji} occurrences of shift j on day i in every employee's schedule. For example, if a particular $n_{ji} = 3$, each employee would work shift j on day i 3 times during the length of the schedule. The total number of days on shift j is denoted by n_{j*} where $n_{j*} = \sum_i n_{ji}$. The total number of working days is denoted by n_* where $n_* = \sum_j n_{j*}$.

Due to the large number of possible schedules, the task of generating feasible shift assignments is not trivial. For a given rotation, with n days on each shift j = 1,2,...,J, there are $\binom{n_*}{n_{1*} \cdots n_{J*}}$ possible schedules. This multinomial expression is at a maximum when the n_{j*} are all equal, ie. when n_{j*} approximates n_* / J for all j. Thus the expression is less than or equal to $(n_*!) / [(n_*/J)! ** J]$, which is of $O(J ** n_*)$. (** denotes exponentiation.) This order of possible schedules can be obtained using the following lower and upper bounds for n! based on Stirling's approximation (these bounds are given in Feller [17]):

$$(2\pi)^{.5}\, n^{n+.5}\, e^{-n+1/(12n+1)} < n! < (2\pi)^{.5}\, n^{n+.5}\, e^{-n+1/(12n)}$$

Substituting the upper bound for the numerator and the lower bound for the denominator in the above expression, it

becomes the following:

$$(2\pi)^{(1-J)/2} \, J^{n_* + .5} \, e^{1/(12n_*) \, - \, J^2/(12n_* + J)}$$

Since $J \geq 1$, this is $\leq J$ ** $(n_* + 0.5)$; because J is independent of n_*, this is O(J ** n_*). Even for two shifts, this figure increases rapidly with the number of working days in a schedule. Therefore, there is a need for procedures that will generate a more meaningful subset, meeting particular objectives, of the total set of schedules.

For the algorithms to be presented here, the following restriction is imposed. The shifts must be labelled such that within a working period (ie. no days off between shifts), for a shift j' to precede shift j, j' must be $\leq$ j. This is a realistic restriction. In most labour agreements of shift workers, there is a stipulation of a minimum number of hours off between shifts. As an example, if three shifts, such as Nights (0001-0800 hrs), Days (0800-1600 hrs), and Evenings (1600-2400 hrs), are used and there is a requirement that a shift on one day must not start earlier than the previous day's shift, then the shifts would be ordered Nights, Days and Evenings. An evening shift, for example, could not directly precede a day shift because there would be less than 24 hours between the start of the two shifts. If such an ordering restriction is not appropriate, the procedures described below could be utilized by considering each valid ordering of the shifts

separately.

If the shifts are assigned in sequence, ie. all of shift 1 before shift 2, etc., then the precedence relationship reduces the number of alternative schedules which need to be considered as described in the following paragraph.

It is useful to define the term "remaining working period". A remaining WP, also abbreviated RWP, is the portion of a WP that has not had a shift assigned to it. Initially, the set of RWP's is the set of WP's. If a remaining WP consists of r days, there are at most r possible ways the shift currently being specified can be assigned to it. These are the first day of the RWP, the first two days, etc. up to all r days. Under a precedence relationship, the current shift must start with the first day of the RWP since otherwise a shift not yet dealt with would precede it. If the number of days assigned is less than r, these days are referred to as a "partial remaining working period".

The five algorithms that will be presented fall into two groups. The first two algorithms exhaustively enumerate all schedules that satisfy the shift precedence restriction described above. The last three algorithms adopt an objective of minimizing the number of shift changes during working period while still adhering to the shift precendence restriction. From the review of the literature and the experience of the Edmonton Police Department, this is a

realistic and meaningful objective. Since the above objective is not the only concern of the workers, the algorithms generate some non-optimal schedules in addition to those with the minimum number of mid-working period shift changes.

Constraints regarding the assignment of shifts can be imposed during the enumeration of the schedules using any of the algorithms. As an example, if a particular shift must be assigned to at least two consecutive days, then this shift would not be assigned to partial RWP's consisting of only one day. As more constraints are added, fewer alternatives are considered during the enumeration which results in faster execution of the algorithms.

The features of the algorithms are summarized below.

Algorithm 3.1: An exhaustive enumeration using a depth first tree generation procedure[29].

Algorithm 3.2: A faster version of Alg. 3.1 that makes use of a property reducing the number of choices during the enumeration.

Algorithm 3.3: A procedure, based on Algorithm 3.2, that generates a subset of all schedules but includes all those with no shift changes during a working period. It also produces other schedules that tend to minimize the number of mid-WP shift schedules. It does not necessarily generate a schedule with the minimum number of these changes if all schedules have shift changes in at least one working period.

---

[29] For a discussion of depth first tree traversing procedures, see Berztiss [9].

Algorithm 3.4: An extension of Algorithm 3.3 that generates more schedules. Although it does not guarantee generation of optimal schedules, stronger statements can be made than for Algorithm 3.3. The schedules produced are not necessarily unique in that a schedule may be generated mcre than once.

Algorithm 3.5: Two rules are added to Alg. 3.4 to reduce the number of duplicate schedules generated.

These algorithms are described in sections 3.4.2 to 3.4.6.

3.4.2 Algorithm 3.1

An exhaustive enumeration using a depth first tree generation procedure can be devised using the property of the ordered shifts. At every point in the enumeration, each RWP with index greater than the previously selected RWP is selected in sequence; the current shift is assigned, if possible, to the first day of the selected RWP, the first two days, etc. up to all r days. This algorithm will be explained using the first shift of the example in Figure 21; this example is a continuation of the one used in the previous section.

Because the enumeration is depth first, the tree represented in Figure 22, which corresponds to the example, is generated in the following order. The shift is assigned to the first day of the first RWP, then to the first day of each of the second, third and fourth RWP's. At this point in the tree, point 'A' in Figure 22, the first day of RWP #5

The working periods obtained from the rotation in Figure 17 are shown below; this rotation is one of the Hamiltonian cycles of the graph G' in Figure 20.

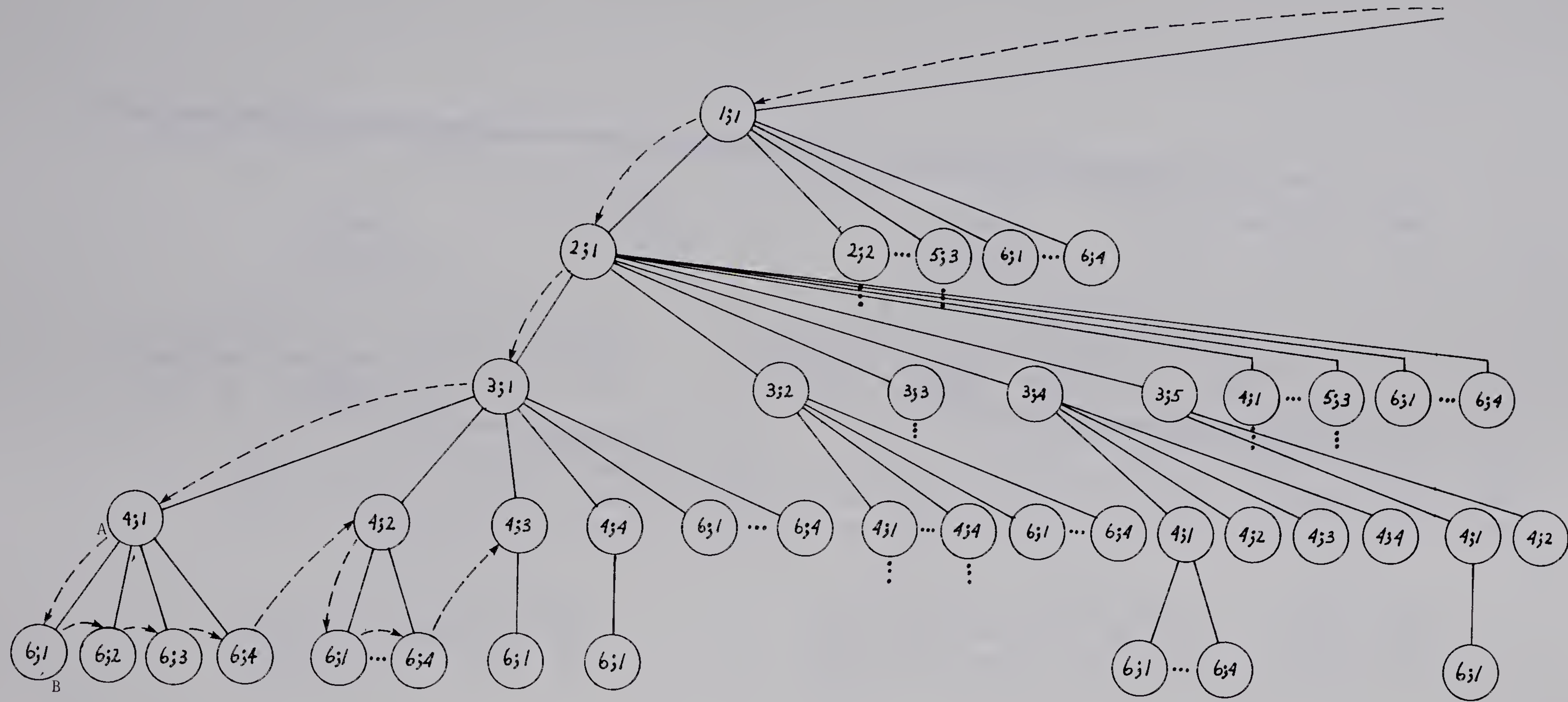
WP #	Working Period	Length
1	Mon-Thu	4
2	Fri-Sun	3
3	Mon-Fri	5
4	Wed-Sat	4
5	Mon-Wed	3
6	Thu-Sun	4

Note: The numbering of working periods is arbitrary.

The manpower allocation for the first shift, taken from Figure 16, is shown below:

	Mon	Tue	Wed	Thu	Fri	Sat	Sun
Shift #1	2	2	2	2	2	2	2

FIGURE 21 - Examples of Working Periods

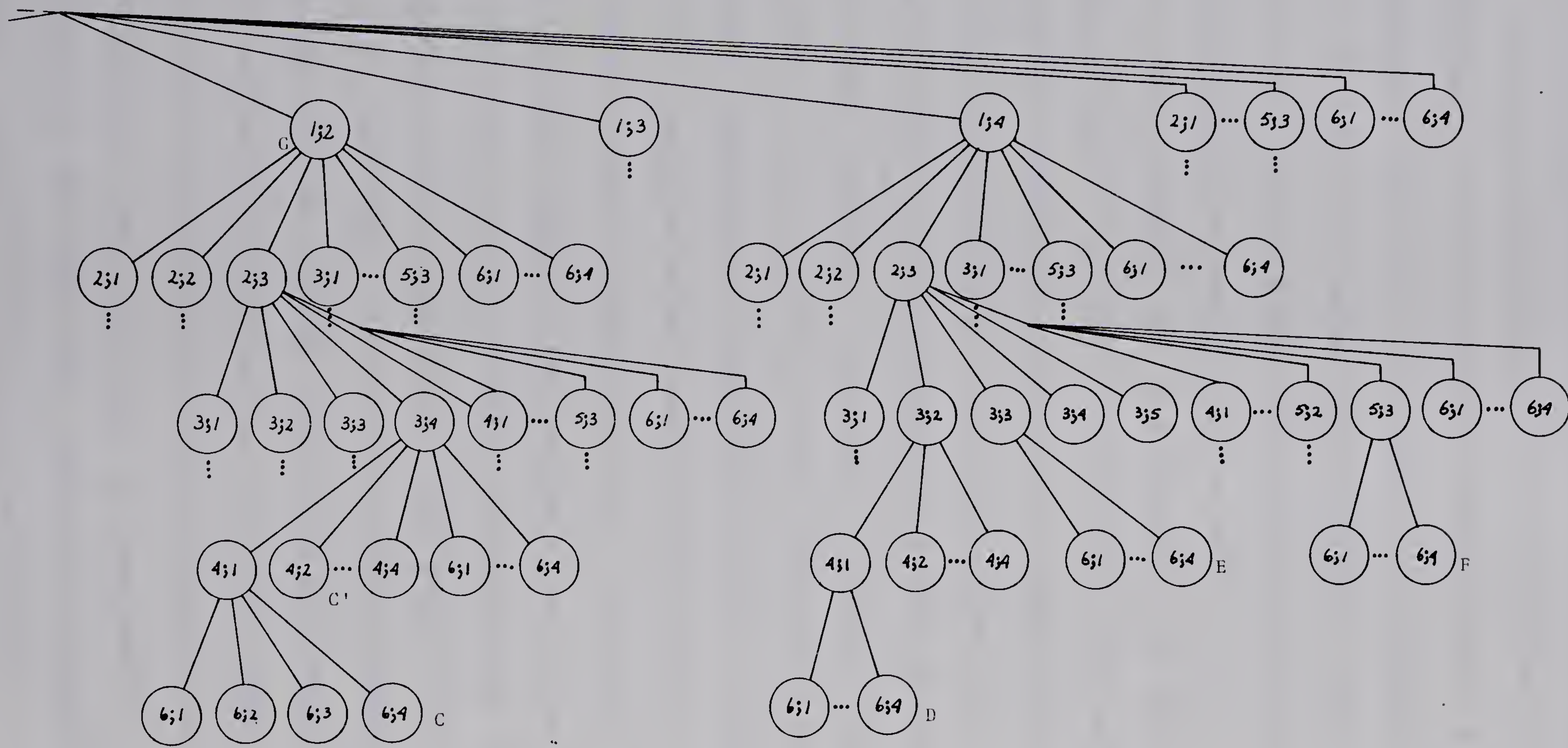


Notation: $i;j$ refers to the first j days of rwp i

••• represents a portion of the tree not shown

Algorithm 3.1 Enumeration Tree

FIGURE 22



Algorithm 3.1 Enumeration Tree (cont'd)

FIGURE 22 continued

cannot be selected because the shift is fully assigned on Mondays. From the original manpower allocation, there are to be 2 occurrences of this shift on Mondays and 2 have already been assigned. Since the shift cannot be assigned to the first day of RWP 5, it cannot be assigned to the first two days, etc. up to all r days. Thus the next node generated is that corresponding to assigning the shift to the first day of RWP 6; this is point 'B' in Figure 22. As there are no more RWP's, this is the end of the path; this path does not correspond to a feasible assignment of this shift because not all the working days on this shift have been assigned. In order for this to be determined, as well as knowing when a day has been completely assigned, a counter must be maintained for each day of the week giving the number of assignments remaining of the current shift to this day. A copy of the counters, which are initially the n_{ji} for the current shift, must be maintained for each path in the tree.

As the last node was the end of a path, the next node at the same level, if one exists, is generated. Thus (6;2) is generated which corresponds to the path (1;1), (2;1), (3;1), (4;1) and (6;2). This path is also terminated because all the days of the current shift have not been assigned. The next two nodes generated are (6;3) and (6;4). These paths are also terminated for the same reason as above. As (6;4) represents that this shift is assigned to the entire RWP 6, which is the last RWP, there are no more nodes generated at this level following node (4;1).

The algorithm then backs up a level and generates the node following (4;1), which is (4;2). The node (5;1) cannot be generated below (4;2) because, as before, the current shift is fully assigned for Mondays. The four nodes generated at this level are (6;1) to (6;4), the same as below (4;1). The order of the tree generated thus far is shown by the dotted lines.

The algorithm continues in the manner described until a feasible assignment of the current shift is found. At this point, if only one shift remains, it is assigned to all unassigned days, which results in a feasible schedule, and the algorithm continues. If more than one shift remains to be assigned, the algorithm is used for the next shift with the new tree rooted at the current node. The same algorithm can be used recursively for this next shift because a new set of RWP's is used consisting of the unassigned days. The above algorithm then continues for the original shift when the trees for the subsequent shifts have been completely traversed. For example, the path ending at node 'C' is a feasible assignment of the first shift; after assigning the second and subsequent shifts, the next node generated for the original shift is 'C1'. Of the complete paths shown, only the nodes marked 'C', 'D', 'E' and 'F' are at the end of paths that result in a feasible assignment of the first shift.

The algorithm is given in Figure 23. For this and the other algorithms, "Pidgeon Algol", as described in Aho et

Let procedure select represent selecting the working periods and procedure assign perform the assignment of the current shift to the working days belonging to the selected working period.

```
begin
procedure select
   begin
   for i <- 1 until 7 do
      begin
      for each WP starting with day i that has
      not been previously selected on this shift do
         assign
      end
   end select

procedure assign
   begin
   i' <- i
   for k <- 1 until length of WP and while there
   are working days on this shift remaining on day i'
   (ie. n_ji'  > 0) do
      begin
      assign this shift to the kth day of this WP
           (ie. day i').
      update the number of working days on this shift
      for day i' (ie. n_ji'  <- n_ji'  - 1)
      i' <- i' ⊕ 1   (addition modulo 7)
      if this shift is completely specified then
         begin
         this shift <- next shift
         if this shift is last shift then
            begin
            assign the last shift to all
                unspecified working days.
            print out schedule
            end
         else
            select
         end
      else
         select
      end
   end assign

comment main procedure
this shift <- first shift
select
end.
```

FIGURE 23 - Algorithm 3.1

al. [2], will be used.

3.4.3 Algorithm 3.2

In Algorithm 3.2, a new rule is used in the enumeration process to more efficiently generate feasible schedules. If the term "sub-RWP" is defined to be either a partial RWP or the complete RWP, then Algorithm 3.1 can be described as selecting all possible sub-RWP's at each point in the enumeration. For Algorithm 3.2, when the number of times the current shift remains to be assigned to each day of the week are not all equal, only those remaining working periods beginning on the first day i where $n_{ji} > n_{ji-1}$ will be selected[30]. If there are no RWP's starting with day i, then the current path in the enumeration is terminated, since no feasible schedules can result. This algorithm is given in Figure 24.

The rationale of the new rule can be expressed as the following property.

Property 1: For each shift j, if there exists a day i where $n_{ji} > n_{ji-1}$, then in order to have a feasible shift assignment, there must exist a remaining working period that begins on day i.

Proof: If there is not a RWP beginning on such a day i, then whenever shift j is assigned to day i, it is also assigned to the previous day, ie. day i-1. Thus, from

[30]All subtractions when referring to days of the week are modulo 7.

Alter procedure <u>select</u> in Algorthm 3.1 to be as follows:

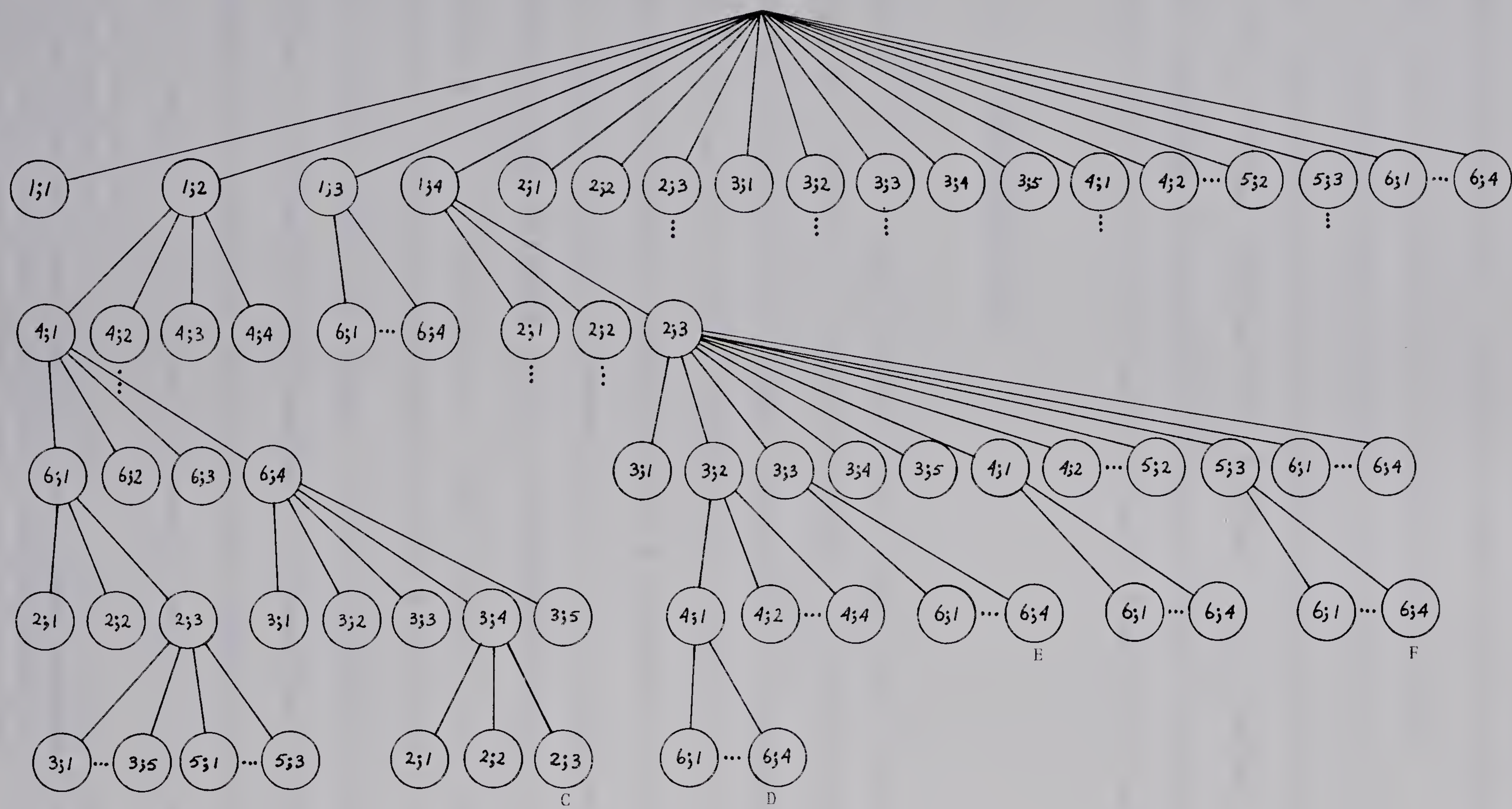
```
procedure select
   begin
   if there exists a day i where n_ji > n_ji-1 for this
   shift then
      begin
      for each WP starting with day i that has
      not been previously selected on this shift do
         assign
      end
   else
      begin
      for i <- 1 until 7 do
         begin
         for each WP starting with day i that is
         not previously selected on this shift do
            assign
         end
      end
   end select
```

FIGURE 24 - Algorithm 3.2

this point in the enumeration, shift j is not assigned to day i more often than to day i-1. This implies $n_{ji} \leq n_{ji-1}$. But this contradicts the premise that $n_{ji} > n_{ji-1}$. Therefore proved.

□

Figure 25 illustrates the application of Algorithm 3.2 to the previous example. The tree is not the same as for Algorithm 3.1 because the remaining working periods are not necessarily selected in the same order. Initially the n_{ji} for the shift in the example equal 2 for all days i; thus the first level of the tree is the same as for Algorithm 3.1. After assigning the current shift to the first day of RWP 1, node (1;1), the n_{j1} for Monday becomes 1. Since the n_{j2} for Tuesday equals 2, only RWP's starting with Tuesday can be selected. As there are no such RWP's, this path is terminated. The next node generated is (1;2). As the n_{j2} for Tuesday also becomes 1, the only RWP that can be selected is the one that starts on Wednesday, RWP 4; thus the node (4;1) is generated below (1;2). Now only RWP's beginning with Thursday can be selected; thus the node (6;1) is next, requiring the next node on the path to start on Friday. There is only one RWP that begins on a Friday, RWP 2. Even though its index (2) is less than the most recently selected RWP (index 6), RWP 2 can be selected because it has not already been dealt with. With Algorithm 3.1 it was dealt with directly below the node (1;2) and in Figure 22, it is



Algorithm 3.2 Enumeration Tree

FIGURE 25

the labelled point 'G'; in Algorithm 3.2, only RWP 4 was selected below (1;2). Thus an index can no lcnger be used to keep track of those RWP's already dealt with.

A RWP has been previously dealt with by Algorithm 3.2 if a node corresponding to it is either on the current path or is an "elder brother" to a node on the current path[31]. This requires maintaining a set of flags for each path indicating which RWP's have already been dealt with.

3.4.4 Algorithm 3.3

Algorithm 3.3, a modification of Algorithm 3.2, only generates schedules with the number of shift changes during a working period at or near a minimum. In Algorithm 3.2, for a RWP of length r, the current shift could be assigned to as many as r sub-RWP's. Algorithm 3.3 assigns the shift only to the longest possible sub-RWP. This is accomplished by assigning the current shift to successive days of the RWP, starting at the first day, until either the current shift cannot be assigned further or until the shift is assigned to the complete RWP. The number of mid-working period shift changes is equal to the number of times the current shift is assigned to a partial RWP, ie. the former case. No mid-WP shift changes result from the shift being assigned to the

[31] An explanation of the familial terminology used for trees can be found in Berztiss[9]. An "elder brother" of a node is a node with the same immediate predecessor (ie. "father") but is traversed earlier then (ie. to the left of) the original node.

complete RWP. Algorithm 3.3 is given in Figure 26.

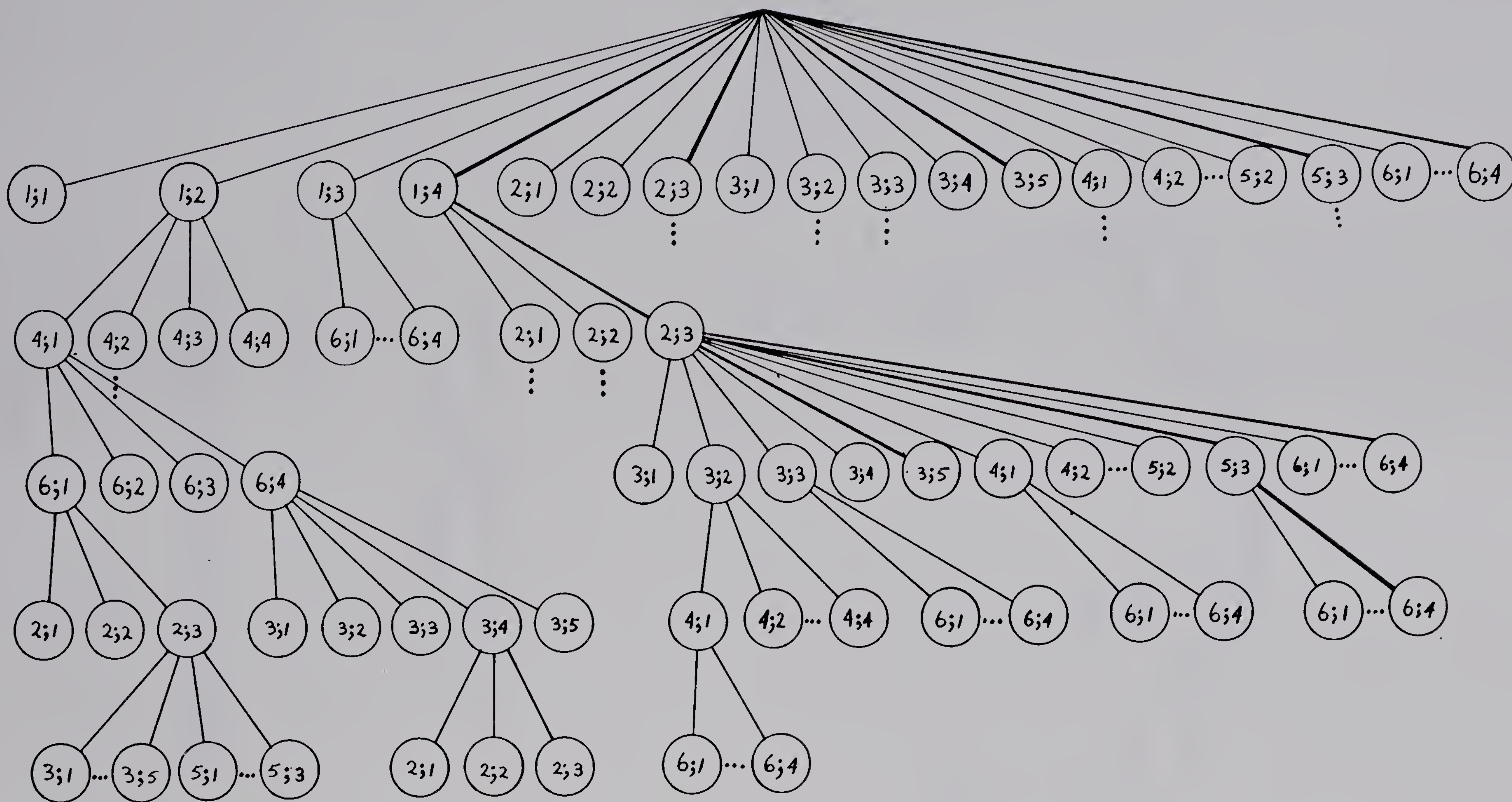
The paths in the tree depicted in Figure 25 that would be generated by Algorithm 3.3 are shown as bold lines in Figure 27. The other paths are those traversed by Algorithm 3.2. The complete tree for the same example using Algorithm 3.3 is shown in Figure 28. The tree is generated in the following manner. Initially, the n_{ji} are equal so each RWP will be selected at this first level. RWP 1 is selected first with the shift assigned to the complete RWP. To continue building this path, the shift must be assigned to a RWP starting with Friday; as RWP 2 is the only one starting on Friday, it is the only one below RWP 1. After this, the n_{ji} are all equal to 1. Thus at this point, each RWP not already dealt with will be selected. First, RWP 3 is selected, but this path is terminated because there are no RWP's starting on a Saturday. RWP 4 is selected, but this path is also terminated because no RWP's beginning with a Sunday exist. RWP 5 is selected next. This path continues because RWP 6 starts on a Thursday; thus RWP 6 is selected. This path represents the feasible assignment of the shift to RWP's 1, 2, 5 and 6. After any subsequent shifts have been assigned, this algorithm backs up one level and RWP 6 is selected below RWP 2. Even though there are three RWP's starting on a Monday, this path terminates because RWP's 1, 3 and 5 have already been dealt with. The algorithm continues until the entire tree is generated.

In the above example, only one feasible assignment was

Alter procedure <u>assign</u> from Algorithm 3.2 to be as follows:

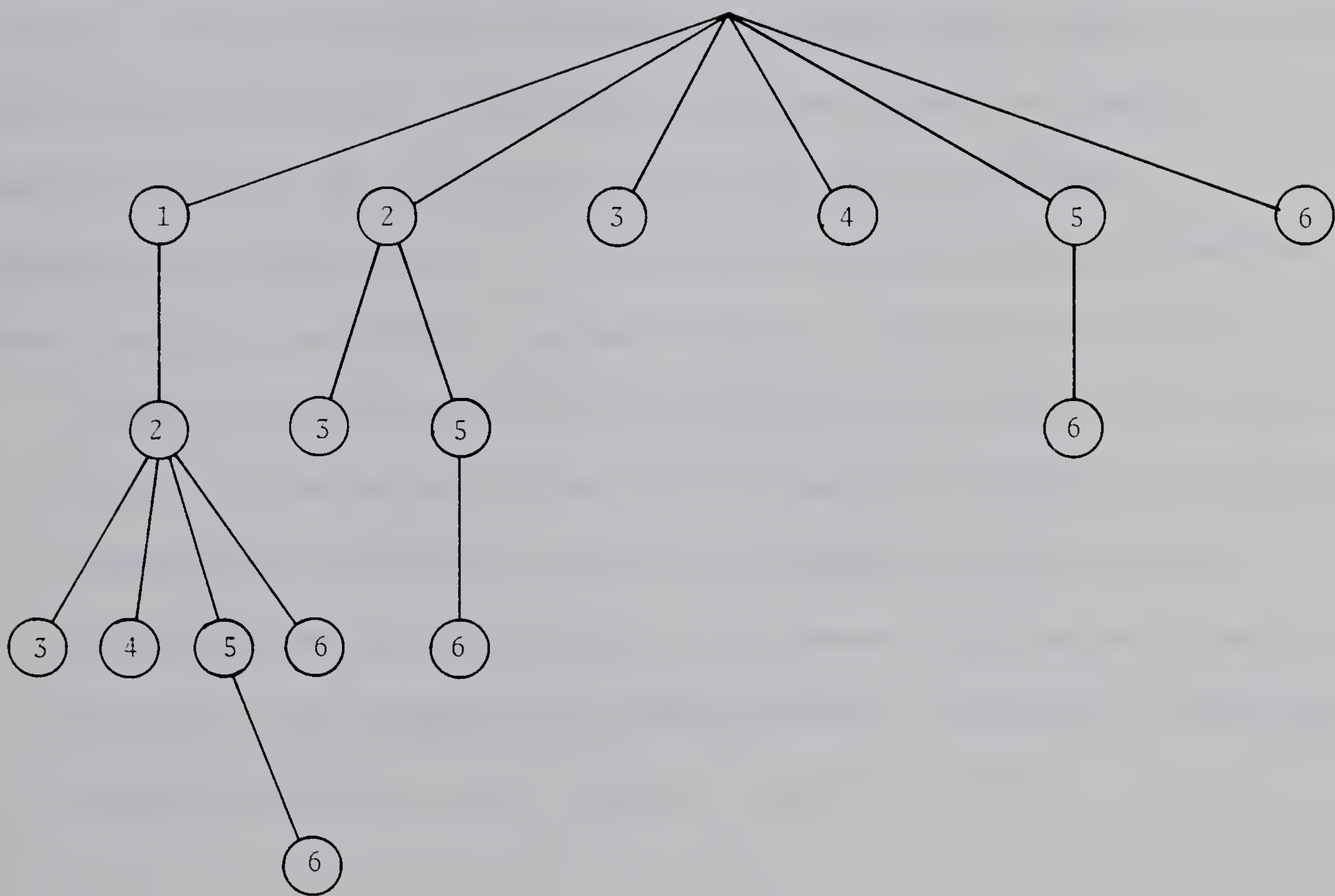
```
procedure assign
   begin
   i' <- i
   for k <- 1 until length of WP and
   while there are working days on this shift
   remaining on day i' (ie. n_ji'  > 0) do
      begin
      assign this shift to the kth day of this WP
          (ie. day i').
      update the number of working days on this shift
      for day i' (ie. n_ji'  <- n_ji'  - 1)
      i' <- i' ⊕ 1   (addition modulo 7)
      end
   comment This part no longer part of the
      loop indexed by k.
   if this shift is completely specified then
      begin
      this shift <- next shift
      if this shift is last shift then
         begin
         assign the last shift to all
             unspecified working days.
         print out schedule
         end
      else
         select
      end
   else
      select
   end assign
```

FIGURE 26 - Algorithm 3.3



Algorithm 3.3 Enumeration Tree (partial)

FIGURE 27



Notation: (i) denotes complete rwp i

1 feasible assignment: 1-2-5-6

Algorithm 3.3 Enumeration Tree (complete)

FIGURE 28

generated. As it consisted entirely of complete RWP's, which were actually the complete working periods, no mid-WP shift changes would have been formed during the assignment of the shift. It was mentioned earlier that Algorithm 3.3 generates those schedules which have the minimum or near-minimum number of mid-WP shift changes. Although this algorithm does not always generate schedules with the minimum number of such changes, the following property can be stated.

Property 2: Algorithm 3.3 generates all feasible schedules that contain no shift changes during a working period.

Proof: Algorithm 3.2 is an exhaustive enumeration. Any path in the enumeration that consists entirely of shifts assigned to complete working periods is also in the enumeration of Algorithm 3.3 because the latter always selects the longest sub-RWP possible. Thus all schedules without mid-WP shift changes are generated.

□

3.4.5 Algorithm 3.4

Algorithm 3.4 differs from Algorithm 3.3 in the following two ways:

1. Remaining working periods are selected that start on all days i where $n_{ji} > n_{ji-1}$. In Algorithms 3.3 and 3.2 only those RWP's that began on the first such day i were selected.

2. Remaining working periods are selected even though they have been dealt with before, providing that they are not on the current enumeration path.

This algorithm is given in Figure 29. The complete enumeration tree for the example used previously is shown in Figure 30. The bold lines represent the enumeration tree using Algorithm 3.3 as shown in Figure 28. Whereas Algorithm 3.3 generated one feasible shift assignment, Algorithm 3.4 generates that assignment and two others along with 13 duplicate assignments.

This algorithm generates more optimal and near-optimal schedules, with respect to minimizing mid-WP shift changes, than Algorithm 3.3 but as shown in the example, it generates some schedules more than once. Although not all optimal schedules are necessarily generated, the following property can be stated.

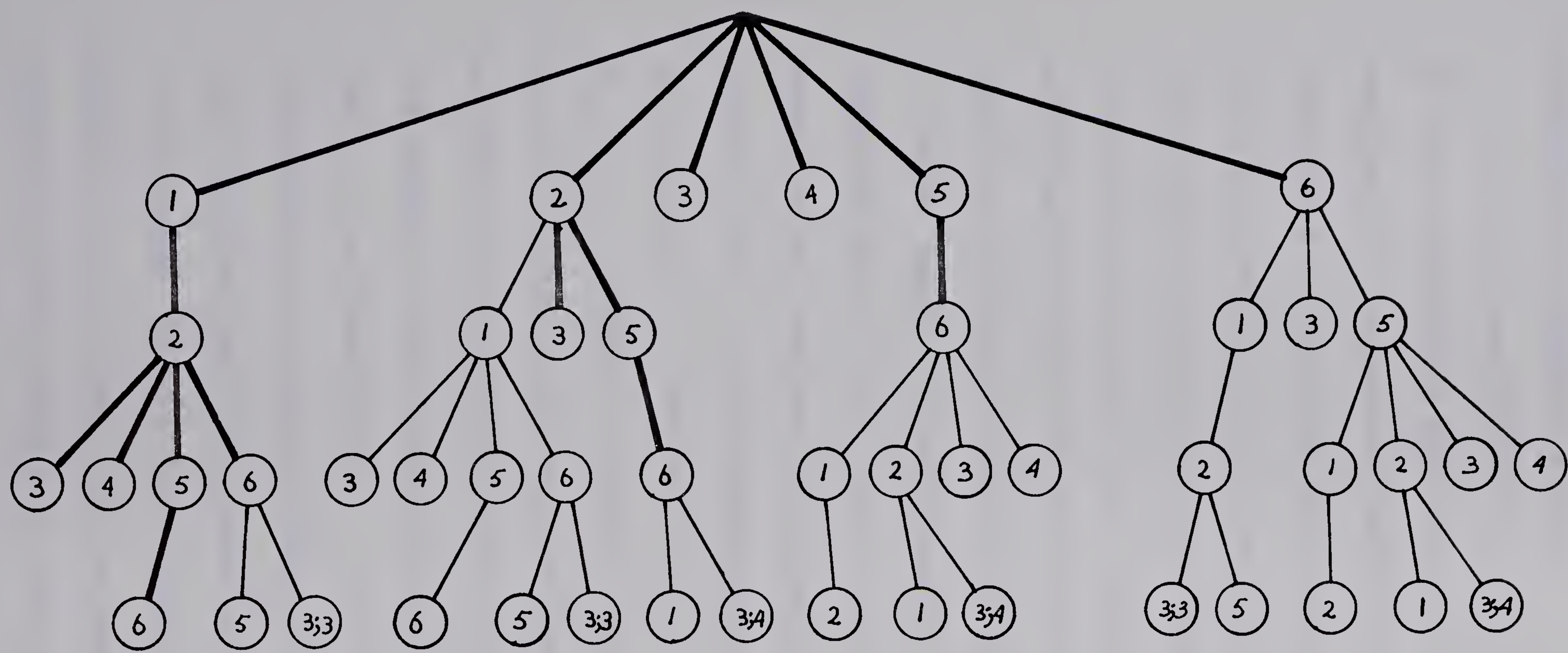
Property 3: For each shift in sequence, Algorithm 3.4 produces all shift assignments containing the minimum number of partial RWP's.

Proof: As this proof is concerned only with the assignment of one particular shift, the term "current shift" refers to the shift currently being assigned. Property 2 showed that Algorithm 3.3, and thus as well 3.4, generates all schedules with all shift assignments to complete working periods. Using the same argument, Alg. 3.4 generates, during the enumeration for each shift, all partial assignments (ie. all partially

Alter procedure <u>select</u> from Algorithm 3.2 to be as follows:

```
procedure select
   begin
   if there exists a day i where n_ji > n_ji-1 for this
   shift then
      begin
      for all days i where n_ji > n_ji-1 for this
      shift do
         begin
         for each WP starting with day i do
            assign
         end
      end
   else
      begin
      for i <- 1 until 7 do
         begin
         for each WP starting with day i do
            assign
         end
      end
   end select
```

FIGURE 29 - Algorithm 3.4



3 feasible assignments: 1-2-5-6
1-2-6-3;3
2-5-6-3;4

13 duplicate assignments

Algorithm 3.4 Enumeration Tree

FIGURE 30

complete assignments) consisting of the current shift assigned to complete RWP's.

For the remainder of this proof, only those paths in the enumeration that do not have the current shift assigned to a complete RWP after the shift has been assigned to a partial RWP will be considered. It is shown in Property 4 in the next section (which is proved independently of the current proof) that restricting the enumeration to these paths does not eliminate any schedule from the set of schedules produced by Alg. 3.4. (This is discussed further in the next section.) Thus, in these paths, all assignments of the current shift to complete RWP's occur first along a path in the enumeration. Because of Property 1, from this point on along a path in the enumeration, the current shift must be assigned to a RWP starting with day i for each case $n_{ji} > n_{ji-1}$ (index j refers to the shift); if $n_{ji} - n_{ji-1} > 1$, the current shift must be assigned $n_{ji} - n_{ji-1}$ times to RWP's beginning with day i. It will now be shown that the current shift will not be assigned to more partial RWP's (from this point in the enumeration after all assignments to complete RWP's) than the minimum number; this implies that this minimum number of partial RWP's is attained. In the proof of Property 4 in the next section, it is shown that no case of $n_{ji} > n_{ji-1}$ is created by the assignment of a shift to a partial RWP using Algorithm 3.4; similarly one such case is reduced

by this assignment. The principal reason for this is that Alg. 3.4 continues to assign a shift to a RWP until it cannot further, either because it has been assigned to the complete RWP or because the shift has been completely assigned for some day resulting in a partial RWP. The first case will not result because all assignments to complete RWP's occurred first in the path in the enumeration tree. Thus the minimum number of partial RWP's determined above is the number that will be generated. All possible assignments with this minimum number of partial RWP's are generated because Alg. 3.4 attempts to assign the current shift to all RWP's that start with all days i such that $n_{ji} > n_{ji-1}$. Thus Alg. 3.4 generates all assignments of the current shift with the minimum number of remaining working periods.

□

Because the shifts are assigned in sequence, a general result concerning the overall minimum number of assignments to RWP's (ie. the number of mid-working period shift changes) cannot be stated. Nevertheless, because the algorithm is performed for each shift except the last one (it is assigned to all days not previously specified), the following corollary can be stated.

Corollary: Algorithm 3.4 generates all schedules with the minimum number of mid-working period shift changes for those scheduling problems containing only two shifts.

3.4.6 Algorithm 3.5

One drawback of Algorithm 3.4 is the number of duplicate schedules produced. Algorithm 3.5 reduces the number of duplicates by using two rules which are described below.

The first rule is that a path in the enumeration is terminated if a particular shift is assigned to a complete RWP after having been assigned to a partial RWP. The rationale is expressed in the following property.

Property 4: Restricting the enumeration to paths where a shift is not assigned to a complete remaining working period after the shift has ben assigned to a partial remaining working period does not eliminate any schedule from the set of schedules produced by Algorithm 3.4.

Proof: It is sufficient to show that for every path that the current shift is assigned to partial RWP's before a complete RWP, there is another path in the enumeration tree with the same sequence except that the current shift is assigned to the complete RWP before the partial RWP's.

If for some day i, $n_{ji} > n_{ji-1}$ after assigning the current shift to a RWP, either $n_{ji} > n_{ji-1}$ before the assignment to that RWP or in that assignment, the current shift was assigned to day i-1 more times than to day i. It will now be shown that after assigning a shift to a partial RWP, the second alternative is never true.

For the second alternative, the only possibility for day i-1 is that it is the last day the shift was assigned to the RWP, ie. the last day of the sub-RWP. However, if this sub-RWP is a partial RWP, then $n_{ji} = 0$. Otherwise, the shift could have been assigned tc day i resulting in a longer sub-RWP. As n_{ji-1} cannot be < 0, n_{ji} is not $>$ n_{ji-1}. Thus cccurrences of $n_{ji} > n_{ji-1}$, for any day i, would have existed before the shift was assigned to the partial RWP. Therefore any complete RWP which is selected after a partial RWP for the same shift, is also selected directly below the previous complete RWP since Algorithm 3.4 selects all RWP's beginning on all days i where $n_{ji} > n_{ji-1}$. This is shown in Figure 31 with the node (e); node (e) follows the partial RWP node (d;y) but also occurs after the previous ccmplete RWP, shown as node (b). If node (b) either is from a previous shift or represents the root, the node (e) would still directly follow it.

It now must be shown that the current shift is assigned to the same partial RWP's after this complete RWP, ie. the nodes (c;x) ... (d;y) are on a path below (e). Let day k be the first day of the complete RWP, ie. node (e). The relation $n_{jk} > n_{jk-1}$ was not changed by the current shift being assigned to each of the partial RWP's. Also, no other relations were changed. Thus assigning the current shift to the complete RWP first would not alter the assignment of the current shift to

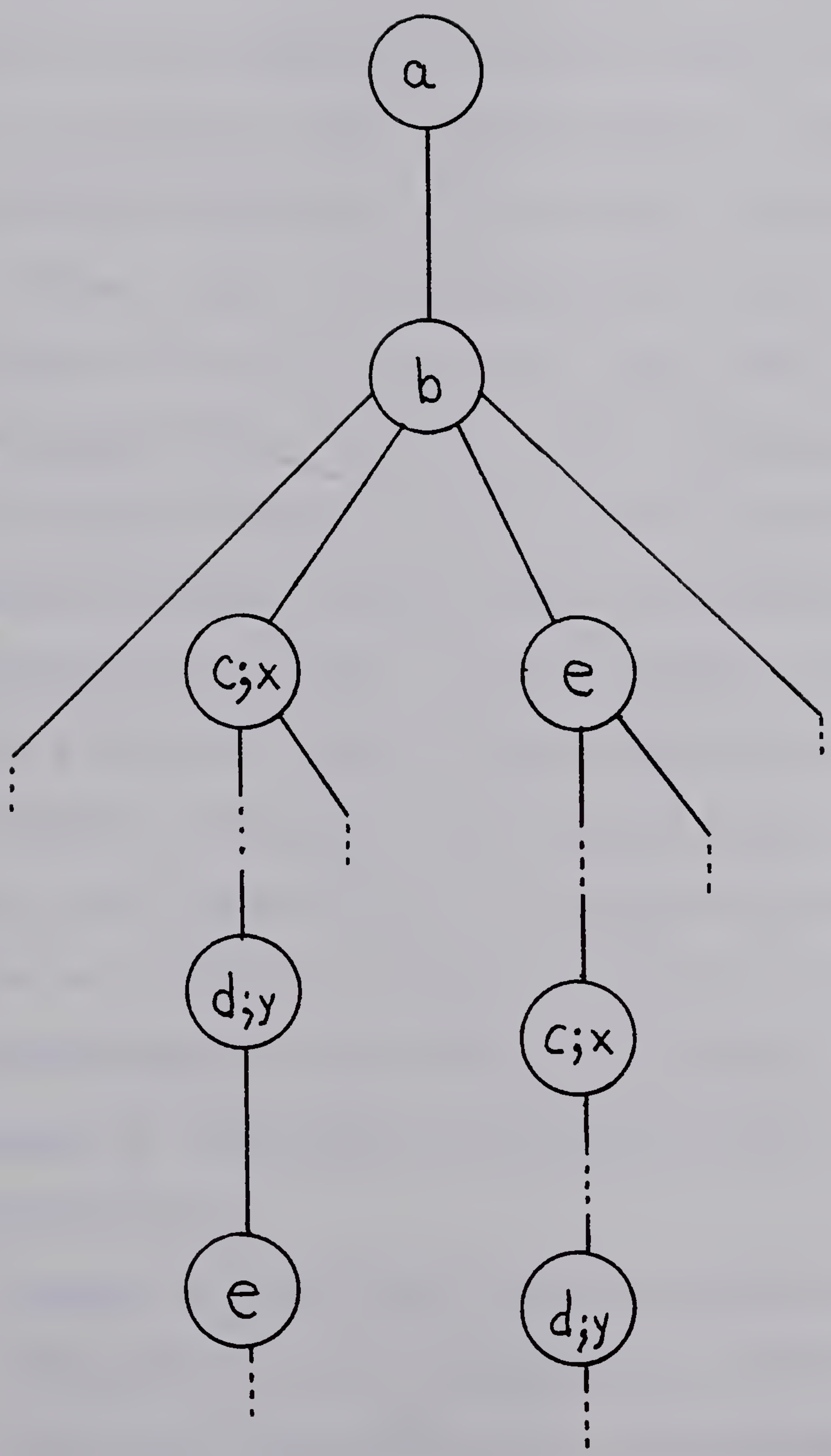


FIGURE 31 - Partial Tree Showing Cutoff

these partial RWP's. Thus another path corresponding to the shift assignment exists. This completes the proof.

□

Before the second rule can be stated, the following definition is necessary. A "free choice point", or fcp, is a point in the enumeration, ie. a node in the tree, where the $n_{ji} > 0$ and are equal for all days i. The "free choice" is that RWP's beginning with any day i can be selected next.

Recall that in Algorithms 3.3 and 3.2, a RWP was not selected if it had previously been dealt with; it has been previously dealt with if a node corresponding to this RWP was either on the path or an elder brother on the path. This pruning rule can be used but in a modified manner. As with Algorithm 3.4, a RWP can be selected if it has already been dealt with but a flag is set noting this. This second rule is stated as follows. If, further along the enumeration path, the current shift is assigned entirely to complete RWP's or a fcp is reached, the path is terminated. The justification is the following property.

Property 5: Algorithm 3.5 generates the same schedules as Algorithm 3.4.

Proof: As the first rule has already been proved in Property 4, it is sufficient to prove that all schedules are still generated using the second rule. The first part of the second rule states that an enumeration path is terminated if the current shift is assigned entirely

to complete RWP's. The proof of this is the same as for Property 2. This proof can also be used for the second part because it can be shown that the current shift is assigned only to complete RWP's on a path leading to a fcp. It was shown in the proof for Property 4 that for the current shift to be assigned to a partial RWP, there is some day i where $n_{ji} = 0$. This implies that no fcp can occur beyond the assignment of a partial RWP, for the current shift, because at a fcp, the n_{ji} are > 0, as well as all equal. This completes the proof.

□

Algorithm 3.5 is given in Figure 32. The enumeration tree for the example used previously is shown in Figure 33; the bold lines represent the paths traversed by this algorithm while the other lines are the arcs traversed by Alg. 3.4 but not be Alg. 3.5.

Alter procedure select from Algorithm 3.4 and procedure assign from Algorithm 3.3 to be as follows:

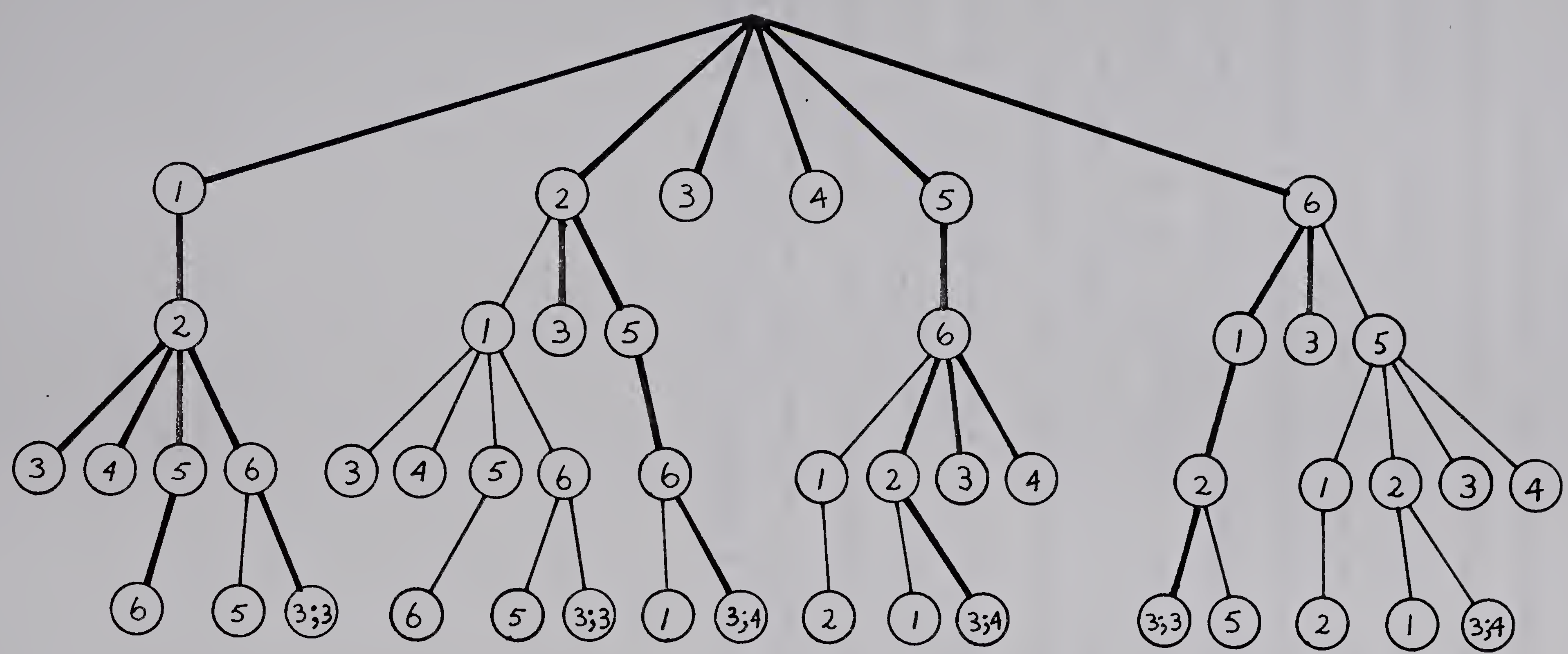
```
procedure select
   begin
   if there exists a day i where n_ji  > n_ji-1  for
   this shift then
      begin
      for all days i where n_ji> n_ji-1for this
      shift do
         begin
         for each WP starting with day i do
            begin
            note if WP previously selected on this
               shift.
            assign
            end
         end
      end
   else
      begin
      comment This is a free choice point.
      if there has not been a WP selected
         that was not previously selected on
         this shift then
         begin
         for i <- 1 until 7 do
            begin
            for each WP starting with day i that is
            not previously selected on this shift do
               assign
            end
         end
      end
   end select


procedure assign
   begin
   i' <- i
   for k <- 1 until length of WP and
   while there are working days on this shift
   remaining on day i' (ie. n_ji'  > 0) do
      begin
      assign this shift to the kth day of this WP
         (ie. day i').
      update the number of working days on this shift
      for day i' (ie. n_ji'  <- n_ji'  - 1)
      i' <- i' ⊕ 1   (addition modulo 7)
      end
```

FIGURE 32 - Algorithm 3.5

```
comment This part no longer part of the
   loop indexed by k.
if this WP partially assigned or no previous
   WP has been partially assigned then
   begin
   if this shift is completely specified then
      begin
      if this WP partially assigned or there has
         not been a WP selected that was previously
         selected on this shift then
         begin
         this shift <- next shift
         if this shift is last shift then
            begin
            assign the last shift to all
                unspecified working days.
            print out schedule
            end
         else
            select
         end
      end
   else
      select
   end
end assign
```

FIGURE 32 continued - Algorithm 3.5



3 feasible assignments: 1-2-5-6
1-2-6-3;5
2-5-6-3;4

2 duplicate assignments

Algorithm 3.5 Enumeration Tree

FIGURE 33

3.5 Beyond Stage 3

Procedures have been described in this chapter which generate feasible shift schedules. The number of schedules produced depends on the algorithms used and the constraints enforced. In the previous section, three algorithms were described that adopted an objective of minimizing the number of shift changes during a working period. However, a single objective is not necessarily consistent with the preferences of workers. If only a few schedules are generated, these can be distributed to the workers for selection. For a large number of schedules, this is not practical. What is needed is some ranking procedure, utilizing multiple objectives, so that only the 'best' few schedules can be identified and distributed. One approach for accomplishing this is outlined as an area for future research in the final chapter.

CHAPTER 4 Discussion of Results

4.1 Introduction and Discussion of Software Used

The Three Stage Method is discussed in two parts. In section 4.2, execution times are given for examples varying several factors for the three stages. In section 4.3, the Three Stage Method is compared with the method of Heller [22, 23, 24].

Throughout this chapter, execution times for the algorithms of the previous chapter as well as for those of Heller [22, 23, 24] are given in central processor (CPU) seconds. These times are for programs run on the Amdahl 470/V6 of the Department of Computing Services of the University of Alberta between January and June 1978. The operating system in use was the Michigan Terminal System (MTS), Distribution 4. Because MTS is a large multi-user, virtual memory time-sharing system, execution times for any particular program and given set of input data tend to fluctuate depending on the load on the system. However, this fluctuation is nominal. Thus CPU times given are for only one run of a program because budgeting restrictions prevented running each program several times for each set of data.

The program for Stage 1 of the method presented in Chapter 3 is written in FORTRAN and uses the revised simplex

method subroutine, ZX3LP, from the IMSL library [26]. The programs for Stages 2 and 3 are written in ALGOL using recursive procedures. Source listings of these programs are contained in Appendix I. The programs for Heller's method are written in FORTRAN and were obtained from N. B. Heller & Associates [24]. The FORTRAN programs were compiled using the FORTRAN IV level G compiler while the ALGOL procedures used the ALGOL W compiler.

For the comparison of the five algorithms for Stage 3, the programs are all similar using the same input procedure, the same data structures and the same basic program structure. Thus the execution times for these algorithms are comparable. The CPU times for the other programs should only be used to compare the effect of different sets of input data for the same program. In addition, CPU times are given for the execution of Heller's method. However, due to differences in the programming implementation of the methods, the execution times of the programs are not comparable.

4.2 Computational Characteristics of the Three Stage Method

4.2.1 Stage 1

Stage 1 is formulated as an integer linear program. However, a revised simplex procedure for ordinary linear programs was used since for all examples, the solutions to the linear program consisted only of integer values. The purpose of this stage is to group the days off into days-off periods. The objective function is to maximize the utility, or benefit, of the set of days-off periods. These same examples, along with the solutions to their first stages, were used as input to the examples for subsequent stages.

The examples ranged from five to fifteen groups of employees to be scheduled; thus the length of the schedules ranged from five to fifteen weeks. These same examples, along with the solutions to their first stages, were used as input to the examples for subsequent stages. The manpower allocations were selected to be as similar as possible. There were two days off per week and the frequencies of each day off differing by not more than one. This is illustrated for two examples in Figure 34. The set of potential days-off periods and their coefficients in the objective function (ie. their preference values) remained the same for all examples. Thus the size of the coefficient matrix never

Manpower allocation for 8 groups:

	Mon	Tue	Wed	Thu	Fri	Sat	Sun
Shift 1	1	2	2	2	2	2	2
Shift 2	2	2	2	2	2	2	1
Shift 3	2	2	2	2	2	2	2
Day off	3	2	2	2	2	2	3

Manpower allocation for 15 groups:

	Mon	Tue	Wed	Thu	Fri	Sat	Sun
Shift 1	3	3	3	3	3	3	3
Shift 2	4	4	4	4	4	4	3
Shift 3	3	4	4	4	4	4	4
Day off	5	4	4	4	4	4	5

FIGURE 34 - Manpower Allocations for 8 and 15 Groups

changed and did not depend on the number of groups. This was reflected in the execution time for the examples being essentially constant, ranging from 0.062 to 0.074 seconds. This stage was also run with examples using different manpower allocations and different coefficients in the objective function. The execution times were also within the above range. This was expected because neither of these two factors affects the size of the linear program. Because this first stage is simply a modification to the formulation by Bodin [10] and the CPU times are much less than those of the other two stages, no further comparisons were performed.

4.2.2 Stage 2

Stage 2 sequences the days-off periods generated by Stage 1 to obtain schedules giving days worked and days off. In Section 3.3, the Stage 2 algorithm was shown to be a tree-based enumeration procedure; therefore it was expected that the execution time would be dependent on the size of the tree. The size of the tree is defined to be the number of arcs. The enumeration tree has as many levels as nodes in the corresponding graph, G' (see Figure 20). There is an arc leading from each node in the tree for each arc leading from the corresponding node in the graph. Thus the size of the tree depends on the size of the graph, defined to be the sum of the number of nodes and arcs. Execution times were compared for examples in which three factors, two of which

directly affecting the size of the graph, were varied. The first such factor selected was the number of groups of people being scheduled, ie. the length of the rotation. As the length of the rotation increased, so did the number of days-off periods, yielding an increase in the number of nodes as well as the number of arcs. The second factor was the combination of constraints on the working period lengths. Clearly the number of arcs increases with the number of acceptable lengths. A constraint placing a non-zero upper limit on the number of working periods of a particular length would not affect the initial size of the graph but would reduce the number of arcs once the limit was reached on any path in the enumeration. The third factor was the manpower allocation; although a change in the manpower allocation did not appear to be directly related to the size of the graph, it was thought that it would have some effect on the size of the enumeration tree and thus on the execution time of the method. These factors are outlined in Figure 35. The examples used for Stage 2 fall into 4 classes. Classes 1 and 2 vary factor 1. Class 3 varies factor 2 while class 4 varies factor 3. The examples used for Stage 2 are summarized in Table 1.

To test the effect of varying factor 1, examples ranging from 5 to 15 weeks were used. These examples were used previously for Stage 1. Working periods of length 4, 5, 6 or 7 days were allowed with no restriction on their frequency. The CPU times are shown in Figure 36 connected by

Factor 1: number of groups (ie. length of schedule)
Factor 1 ranges from 5 to 15 groups.

Factor 2: constraint combination
There are 4 levels of factor 2:

1) no limit on working periods (WP's) of length 4, 5 & 6; no more than 1 of length 7.

2) no limit on WP's of length 5, 6 & 7; no WP's of length 4.

3) no limit on WP's of length 5, 6 & 7; no more than 1 of length 4.

4) no limit on WP's of length 4, 5, 6 & 7.

Working periods shorter than 4 days and longer than 7 days are not permitted.

Factor 3: manpower allocation
There are three alternatives for factor 3:

Case	Mon	Tue	Wed	Thu	Fri	Sat	Sun	Total
a	5	4	4	4	4	4	5	30
b	5	5	4	4	3	4	5	30
c	5	5	4	4	3	3	6	30

The range of frequencies of days off for cases a, b and c is 1, 2 and 3, respectively.

FIGURE 35 - Factors for Stage 2

Class	Example Conditions	Results Given in
1	Factor levels are (*,4,a)+ Factor 1 varies from 5 to 15 groups.	Figures 36 and 37
2	Factor levels are (*,1,a). Factor 1 varies from 5 to 15 groups.	Figures 36 and 37
3	Factor levels are (15,*,b). Factor 2 varies from constraint combination 1 to 5.	Table 2
4	Factor levels are (15,4,*). Factor 3 varies from allocation a to c.	Figure 38

+ Factor levels refer to factor values given in Figure 35. * refers to the level being varied for the examples in the class.

TABLE 1 - Summary of Examples - Stage 2

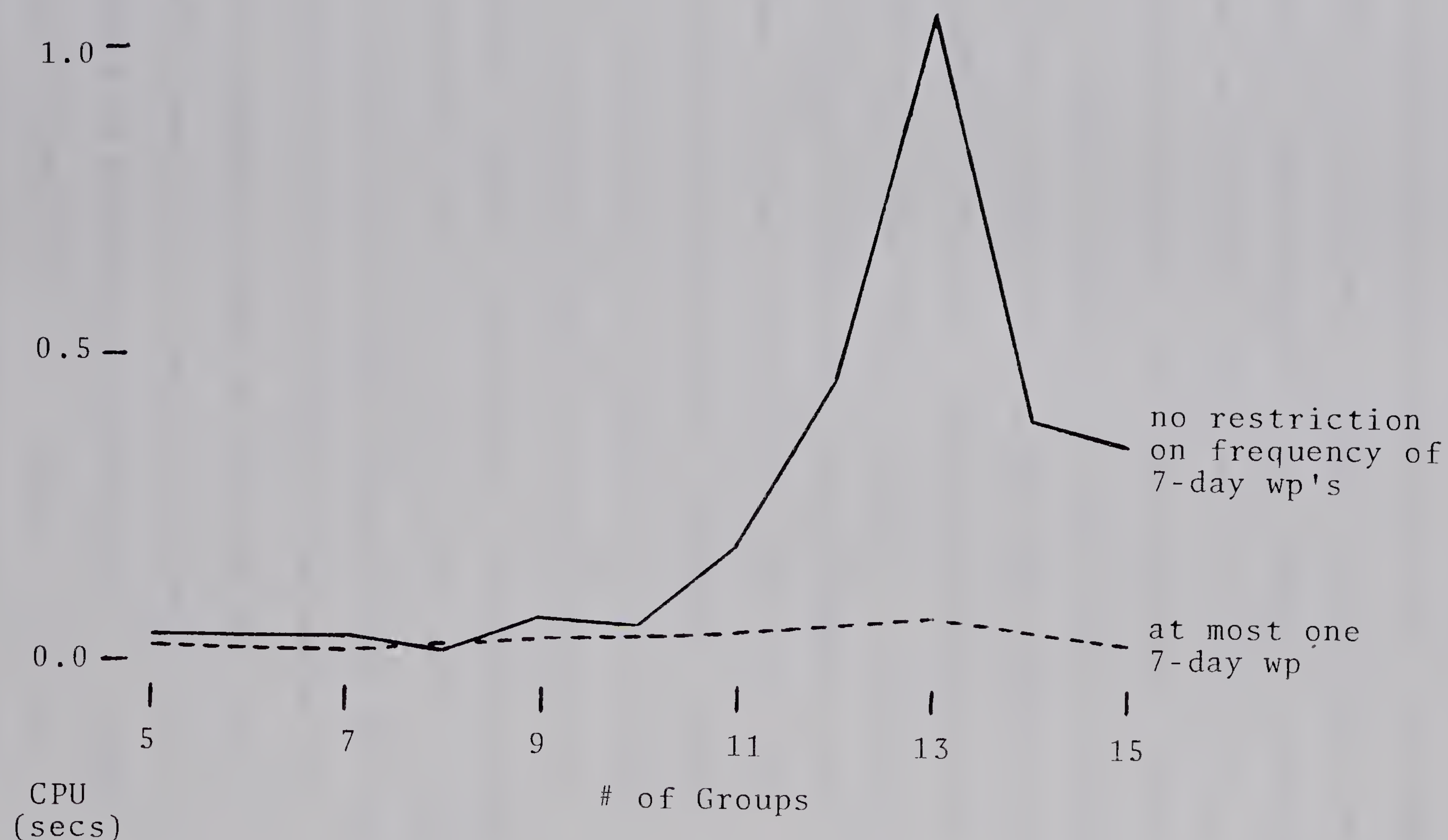


FIGURE 36 - Execution Time for Stage 2

solid lines. Although there appears to be somewhat of an upward trend with an increasing number of groups, there is not an obvious relationship with execution time. The number of rotations generated follows a similar relationship, as shown in Figure 37. The variation in the number of rotations and CPU time can be partially explained by noting that as the number of groups changed, so did the number of working days, the set of days-off periods and the relative frequency of working each day of the week (eg. for some examples there were the same number of Fridays worked as Thursdays while for others, there were more Fridays worked.). Although the examples were similar to each other, as mentioned in the previous section, it is not possible for the number of groups to vary with all other factors remaining constant. The number of working days is an example of one of these other factors that explains some of the variation. The number of rotations depends partially on the number of ways the number of working days can be partitioned into feasible working period lengths; this number of partitions does not necessarily increase with an increase in the number of working days. For example, there is only one way that 30 days can be partitioned into 5 periods of length 4, 5 or 6: 6,6,6,6,6, whereas 25 days can be partitioned in 3 ways: 5,5,5,5,5, 6,5,5,5,4 and 6,6,5,4,4. Because the Stage 2 procedure is executed once for each rotation, it was expected that execution time would be a function of the number of rotations. This is supported by the similarity of

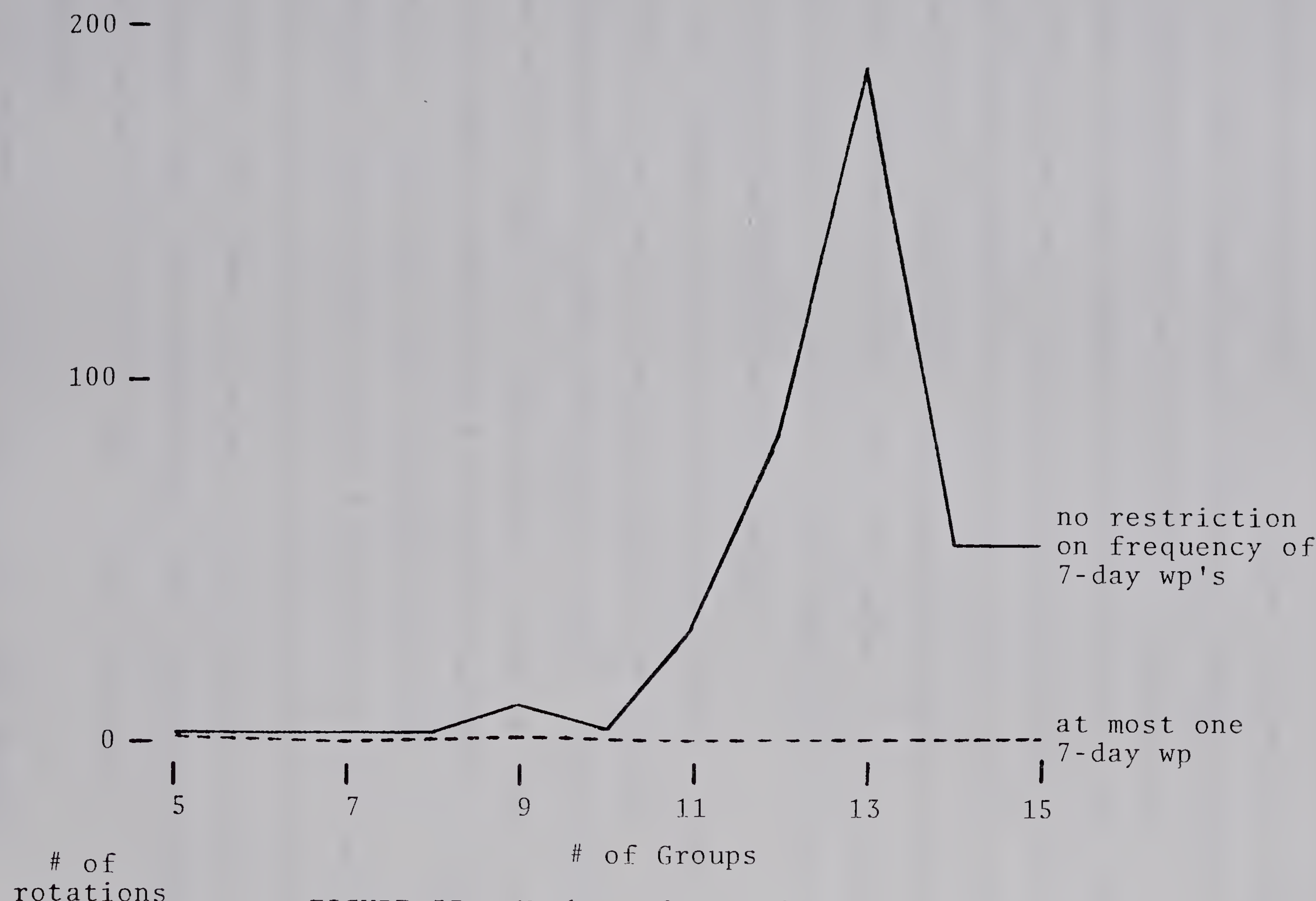


FIGURE 37 - Number of Rotations - Stage 2

the relationships for execution time and number of rotations shown in Figure 36 and 37. Thus an increase in the number of partitions should result in an increase in execution times.

In order to observe the effect of altering factor 2, ie. constraining the set of allowable working period lengths, the above examples were rerun with the constraint of allowing not more than one 7-day working period. The CPU times for the examples of class 2 are shown in Figure 36 connected by dashed lines. The execution times are much less than for the previous examples; the number of rotations, as shown in Figure 37, as well, is much less with no more than one rotation generated for any example.

The effect, on both execution time and the number of rotations generated, of varying factor 2 is further demonstrated by the four examples of class 3. The execution times and the number of rotations are given in Table 2. For the case with no restrictions on working periods of length 4 through 7, there were 1722 rotations generated, an unfeasible number since a Stage 3 algorithm is run for each rotation. On the other hand, no rotations were generated for the other three cases. In order to feasibly solve this Stage 2 problem, cases more constraining than the first case but less constraining than the other three would have to be attempted.

The effect of varying factor 3, ie. the manpower allocation was observed for the three examples of 15 groups in class 4 (see Table 1). As the range of the the

The following table gives the execution time and number of rotations generated for four cases of 15 groups. The maximum frequency of working periods of length 4 and of length 7 are given in the table. There are no constraints on the frequencies of working periods of length 5 or 6. Working periods of 3 days or less or 8 days or more are not allowed.

Upper limit of length		CPU	# of
4 days	7 days	Time	Rotations
no limit	1	0.24	0
0	no limit	0.58	0
1	no limit	0.93	0
no limit	no limit	9.98	1722

TABLE 2 - Effect of Working Period Length Constraints

frequencies of days off increased, so did the CPU time and the number of rotations, as shown in Figure 38.

The results of this section illustrate that the number of groups being scheduled (factor 1) has a moderate influence on the size of the Stage 2 problem. On the other hand, the restrictions on the lengths of acceptable working periods (factor 2) and the manpower allocation (factor 3) greatly affected the execution times. The more restrictive the upper limits for frequencies of these lengths, the faster the rotations were generated. Thus, if a particular length of working period was acceptable but not very desirable, it would be advantageous initially to not include this length in the set of allowable lengths. If no feasible rotations are generated, then this length can be included such that it occurs once or not at all; this upper limit can be gradually increased until rotations are produced. By adopting this approach, the execution time and the number of rotations generated should be reduced relative to not adding restrictions initially. The manpower allocation is specified prior to the use of this shift scheduling method and even though it is a fixed input, it could be used to give an indication as to the size of the problem. It was observed that with a greater range in the frequencies of each day of the week being a day off, the larger the problem. For these larger problems, restrictions on the set of allowable working period lengths are more important for reducing the execution time.

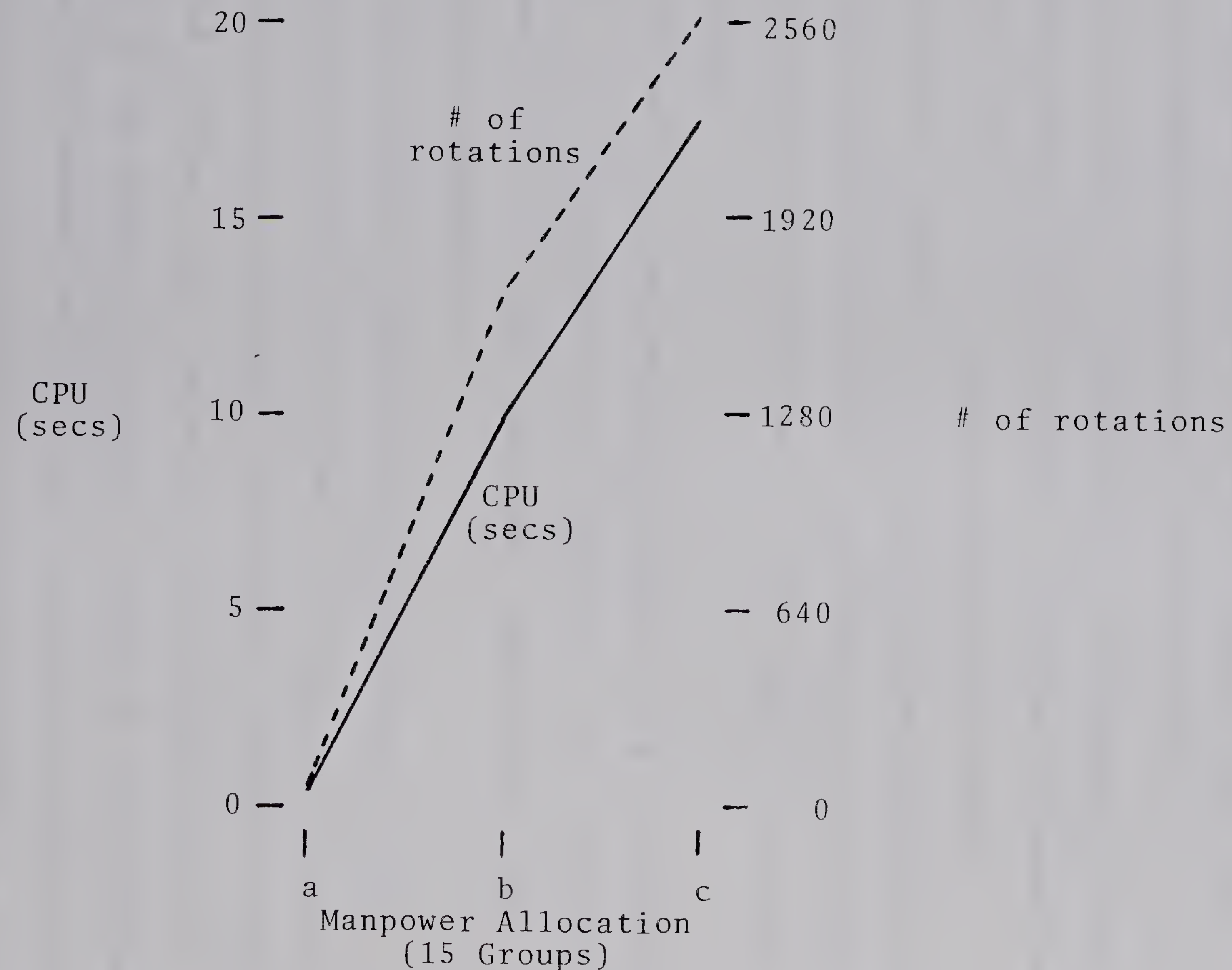


FIGURE 38 - Execution Time & # of Rotations

4.2.3 Stage 3

In Chapter 3, five algorithms were described for Stage 3, the assignment of shifts to working days. Algorithms 3.1 and 3.2 are complete enumerations. Algorithms 3.3, 3.4 and 3.5 adopt the objective of minimizing the frequency of shift changes during a working period. Algorithm 3.3 generates all schedules with no mid-working period shift changes. Algorithm 3.4 generates more schedules than Algorithm 3.3 but also produces duplicates. Algorithm 3.5 incorporates two pruning rules into Algorithm 3.4 that reduces the number of duplicates.

The factors that varied for Stage 3 are the number of groups (ie. the length of the schedule), the constraint combination and the algorithm used. These factors are outlined in Figure 39.

Two classes of comparisons are discussed in this section. The first class consists cf comparing execution times of each for the five algorithms for several example cases of similar size. The second class deals with the effect of the number of scheduled groups on the execution time of the fastest method, Algorithm 3.3. For both classes mentioned above, the effect of adding constraints regarding the form of acceptable schedules is also discussed. These examples are summarized in Table 3.

The CPU times and the number of schedules generated for

Factor 1: number of groups (ie. length of schedule)
Factor 1 ranges from 5 to 15 groups.

Factor 2: constraint combination
Five combinations of the following three constraints are used.

1) Night shift may not follow Afternoon shift, even with intervening days off, without an intervening Day shift.
pt> No Day shift may be assigned to a working period preceding a weekend off. A weekend off is a days-off period that includes Saturday and Sunday.

2) No Day shift may be assigned to a working period following a weekend off.

The 5 levels of factor 2, ie. the five combinations, are:

1) constraints 1, 2 & 3.

2) constraints 1 & 2.

3) constraints 2 & 3.

4) constraint 1.

5) no constraints

Factor 3: the Algorithm used.
Factor 3 ranges from Alg. 3.1 to 3.5.

FIGURE 39 Factors for Stage 3

Class	Example Conditions	Results Given in
1	Factor levels are (9,*,*). Factor 2 varies from constraint combination 1 to 5. Factor 3 varies from Alg. 3.1 to 3.5. In order to generate more examples, Stage 1 was run with two objective functions which generates two sets of DOP's (A & B). Set A has one 4-day and one 2-day weekend off; Set B has two 3-day weekends off. Stage 2 generates a set of rotations for each set of DOP's. These two sets, in conjunction with the 5 constraint combinations are labelled cases A1 to A5 and B1 to B5.	Tables 4, 5 Figure 40
2	Factor levels are (*,*,Alg. 3.1) Factor 1 varies from 5 to 15 groups. Constraint combinations 1, 4 & 5 are used.	Figure 41

TABLE 3 - Summary of Examples - Stage 3

the ten cases in class 1 are given in Tables 4 and 5, respectively. The logarithms to the base 10 of the CPU times per rotation are graphed in Figure 40; logarithms reduce the range of values and hence permit a graphical comparison between the fastest algorithm (3.3) and the slowest (3.1). The execution times are expressed per rotation because of the differing number of rotations. (5 for cases A1-A5 and 7 for cases B1-B5). Comparing the algorithms that enumerate all feasible schedules, 3.2 executes quicker than 3.1 which reflects the reduction in the size of the enumeration tree, as discussed in Section 3.4. Algorithm 3.2 was able to solve some problems for which 3.1 took an unrealistic length of time; an example of this is case B4 for which 3.2 generated 38 schedules in 41.78 seconds whereas 3.1 had generated only one schedule after 80 seconds and still was not finished executing. However there were cases, those with none of the constraints applied, for which Algorithm 3.2 did not produce a feasible solution. As expected, the execution time increased as fewer constraints were applied.

The three algorithms that generate a subset of feasible schedules are much faster than the complete enumerative algorithms. Algorithm 3.3 executes more quickly than either 3.4 or 3.5 but did not generate feasible schedules for three cases (A1, A2 and A4). The number of schedules generated for the other cases was small enough that the schedules produced could be distributed to employees for selection. On the other hand, the number of schedules generated by 3.4 and 3.5

	Algorithm				
Case	3.1	3.2	3.3	3.4	3.5
A1	59.66	12.63	0.12	0.36	0.39
A2	153.18	26.2	0.13	0.49	0.57
A3	**	23.27	0.15	0.71	0.70
A4	**	59.82	0.12	0.96	1.07
A5	**	**	0.17	4.05	3.57
B1	53.34	9.61	0.12	0.38	0.37
B2	**	**	0.12	0.51	0.52
B3	59.14	17.82	0.15	0.74	0.66
B4	*	41.78	0.12	0.73	0.77
B5	**	**	0.15	3.42	2.63

* - 1 schedule after 80 seconds

** - not run because of budget constraints

TABLE 4 - CPU Times for Cases A1 - B5 for Stage 3

	Algorithm				
Case	3.1	3.2	3.3	3.4	3.5
A1	4	4	0	2	2
A2	15	15	0	6	6
A3	**	840	4	40	32
A4	**	47	0	24	21
A5	**	**	5	300	255
B1	12	12	2	8	8
B2	**	**	2	11	11
B3	700	700	4	46	37
B4	**	38	2	18	18
B5	**	**	6	250	204

** - not run because of budget constraints

Note: The numbers include duplicate schedules generated.

TABLE 5 - # of Schedules for Cases A1 - B5 for Stage 3

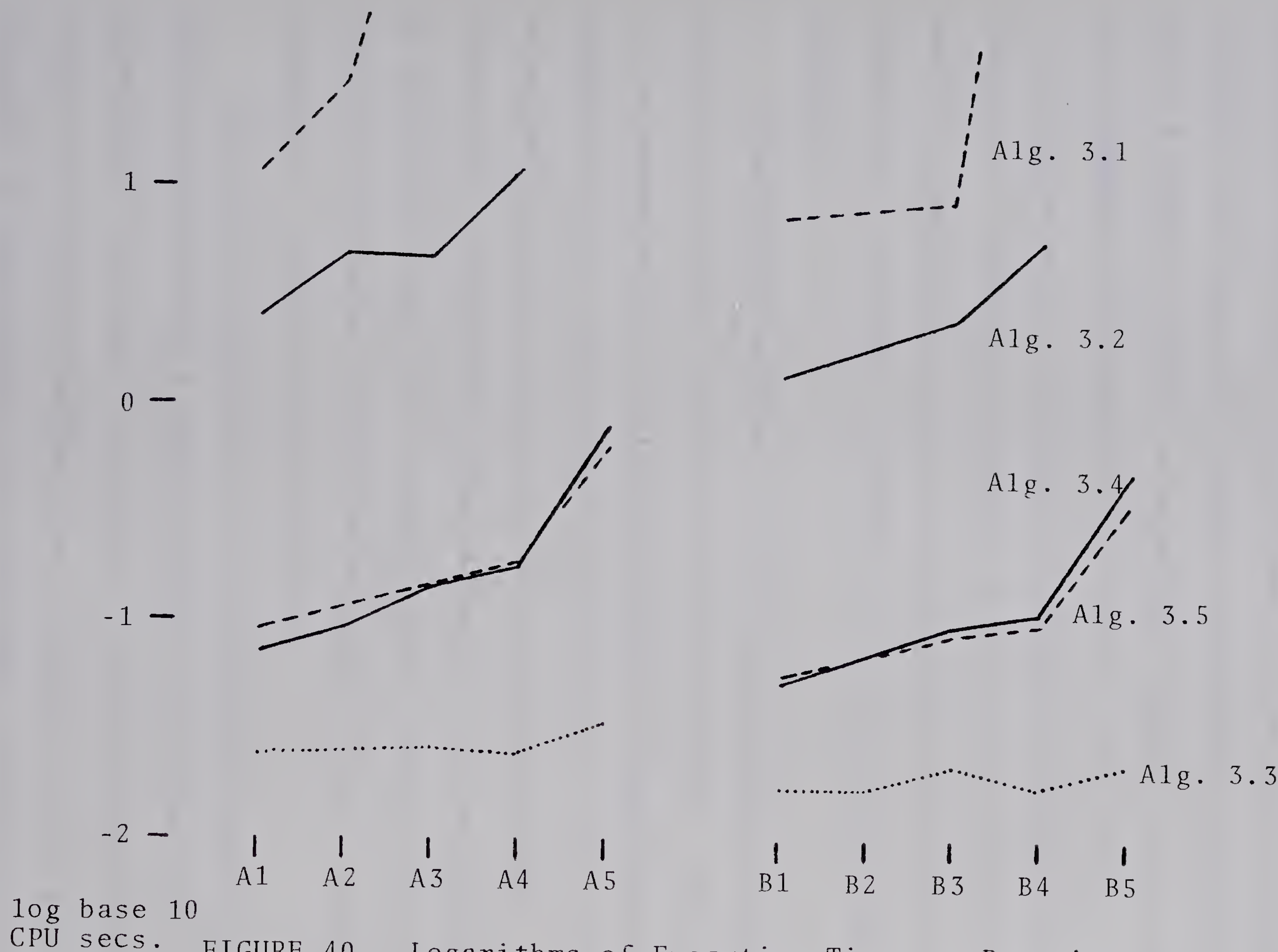


FIGURE 40 - Logarithms of Execution Time per Rotation
Algorithms 3.1-3.5

for most cases was sufficiently large to demand some selection procedure for reducing the set of alternative schedules. (This topic is discussed in the final chapter.) However, for cases with the most restricitve constraint sets, A1, A2, B1 and B2, a manageable number of schedules was produced. This supports the approach of initially enforcing as many constraints as possible and then relaxing them until a desirable schedule is found. For Algorithms 3.4 and 3.5, the execution time increases with the relaxation of constraints, but with Algorithm 3.3, little change was observed.

As discussed earlier, the difference between Algorithms 3.4 and 3.5 is that the latter is able to prune more paths in the enumeraticn to reduce the number of duplicates. This reduction in the total number of schedules is also reflected in execution times, especially for the cases with no constraints, A5 and B5. Hence, Algorithm 3.5 is definitely an improvement over 3.4, because for the larger cases, CPU time and the number of duplicates generated are reduced, and at worst, it is not slower than 3.4.

Adopting the objective of minimizing the number of shift changes during a working period allows much faster algorithms to be used for Stage 3. If eliminating such shift changes entirely is desired, then Algorithm 3.3 is most suitable. If some changes are acceptable, Algorithm 3.5 can be used; however, if few constraints are applied, a better approach would be to use Algorithm 3.3 and then, if no

suitable schedules were produced, to use Algorithm 3.5. If minimizing these mid-working period shift changes is not desired, the above two algorithms are not suitable; if several constraints are applied, Algorithm 3.2 could be used but the execution times would be much longer than those of either Algorithms 3.3 or 3.5.

The class 2 examples were designed to observe the effect of the number of groups being scheduled (ie. the length of the schedule) on execution time where only Algorithm 3.3 was considered. As with class 1, execution times are expressed in CPU time per rotation. The results of all three constraints are graphed in Figure 41 using logarithms of the above measure. Results for 13 and 14 weeks with the second constraint set and for 12, 13 and 14 weeks with the third constraint set are not given. After 10 seconds of execution, very few, and sometimes none, of the rotations had been completed. For the results of the first constraint combination, only 5 of the 54 rotations for 14 weeks were used and for the second combination, only 10 of the 86 rotations for 12 weeks were used.

The execution time varies considerably across the different examples for each of the three constraint sets; the times range from 0.025 seconds per rotation to over 10 seconds per rotation. There is a slight increasing trend with increasing length of schedules. The variation cannot be fully explained but must be partially due to the example cases differing by more than just the number of groups. As

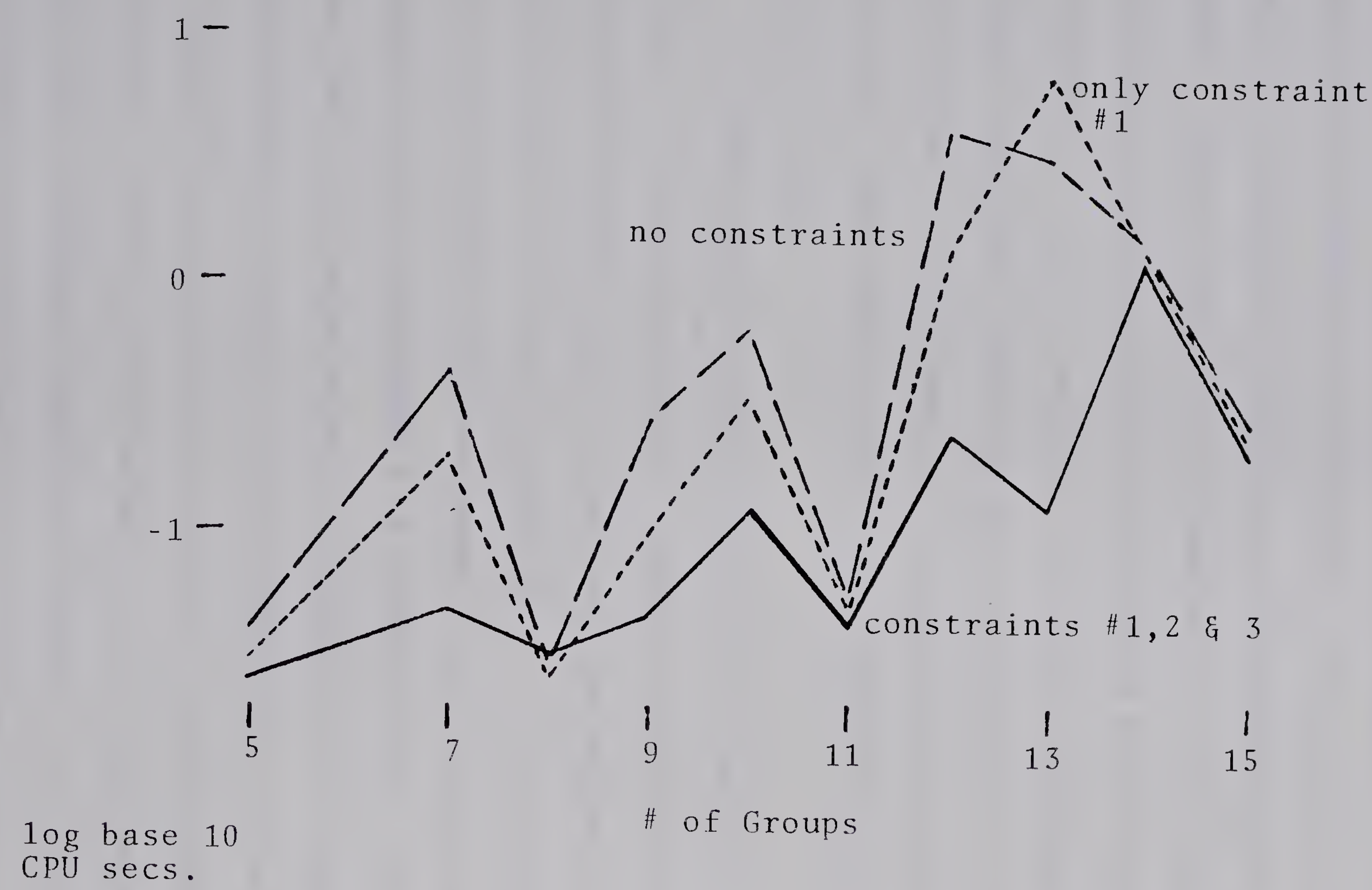


FIGURE 41 - Logarithms of Execution Time per Rotation - Algorithm 3.3

discussed in the previous section, the manpower allocation and the set of dop's change, which results in rotations containing different combinations of lengths of working periods. Different rotations would encounter free choice points (fcp's) during the enumeration with differing frequency. Execution time is believed to be dependent on the frequency of fcp's, because at an fcp, remaining working periods beginning with any day of the week are attempted. At other points, only those beginning with one particular day are used. (Free choice points are defined and discussed in Section 3.4.6)

In some cases, the relaxation of constraints has little effect on CPU time while in others, the execution time increases dramatically. A table of the number of schedules per rotation is given in Table 6; it is noted that the cases for which the CPU time increased significantly with the relaxation of constraints had significantly more schedules generated. For these cases, the approach of applying as many constraints as possible has two benefits: the execution time to generate the schedules is reduced and the process of selecting a small set of schedules is facilitated by having much fewer schedules produced. The use of a similar approach for Stage 2 is also important. The above comparisons were made on a "per rotation" basis; fewer rotations would mean less execution time and fewer schedules generated for selection.

With the first class of comparisons, the relative speed

Algorithm 3.3 is run for test case of from 5 to 15 groups for three constraint sets. The constraints are given in the text. The three sets are all three constraints (1&3), only the first constraint (1) and none of the three constraints (none). The follcwing table gives the number of schedules produced for each case along with the number of rotations used.

# of groups	# of rotations	# of schedules		
	Constraints:	1&3	1	none
5	2	1	2	6
7	2	0	24	78
8	2	0	0	0
9	9	0	11	176
10	2	0	14	208
11	32	1	1	6
12	86	33	297*	
13	186	24		
14	54	2**		
15	54	0	0	0

* - Only 10 rotations used

** - Only 5 rotations used

TABLE 6 - Number of Schedules using Algorithm 3.3

of five algorithms for Stage 3 was demonstrated. For all but Algorithm 3.3, the relaxing of constraints resulted in significant increases in execution time. With the second class of examples, no strong relationship between execution time and number of groups being scheduled was observed, as shown in Figure 41. The relaxing of constraints dramatically increased CPU time for some cases but not for others; once again no relationship was found. Because of the observed effects of constraints, the following approach is recommended: Translate schedule properties desired by employees into constraints; transform continuous objectives, such as minimizing shift changes, into constraints restricting the value at or near an optimal value. If no acceptable schedules are generated with all constraints applied, then constraints should be gradually relaxed until acceptable schedules are produced. As has been demonstrated, there are some problems for which the Stage 3 algorithms require an unreasonable amount of time. If constraints cannot be added, then some other method must be used. However, many realistic problems are able to be solved by the method, as shown in the examples. The usefulness of the method on a real problem is demonstrated in Chapter 5 with a discussion of an application of the Three Stage Method in the Edmonton Police Department.

4.3 Comparison of the Three Stage Method with Heller's Method

In this section, the three stage shift scheduling method described in Chapter 3 is compared with the method of Heller et al. [22, 23, 24], which is described in section 2.2.3. These comparisons are based on the types of scheduling problems that can be solved and on the the types of schedules generated rather than on their respective computational efficiency. The two methods, although similar, are best suited to different problems and therefore a comparison of execution times will not be performed. Heller's method was chosen because it was the only enumerative, cyclic shift scheduling method described in the literature and it has been widely implemented, mainly in police departments. In the Three Stage Method, the first stage determines the days-off periods by optimizing with respect to a utility function incorporating employee preferences. This set of days-off periods is then fixed for all schedules that are generated. The second stage generates rotations of these days-off periods in a manner similar to the generation of single-shift schedules by Heller's procedure except that no objective function is used. The third stage then generates completed schedules by specifying the shifts to be worked on the working days. In Heller's method, the shifts are already specified and the remaining task is to select the single-shift schedules to comprise the multi-shift schedules.

One principal difference between the two methods concerns the "structure" of the schedules produced. The schedules produced by Heller's method can be described as having the same structure because they consist of blocks of consecutive weeks, one and only one block for each shift. Heller's method fixes the structure of the schedule and all combinations of days-off periods are enumerated whereas the Three Stage Method determines the set of days-off periods at the first stage and does not restrict the structure. If a suitable schedule is not found with one set of days-off periods, another set, determined by the first stage with a revised objective function, can be used. However, a similar capability does not exist with Heller's method becasue the procedure is based on the fixed schedule structure. Heller suggests an approach for modifying the schedules by splitting a shift into two or more parts but no procedure is given. This approach is elaborated upon later in this section.

Because of the structure of the schedules produced by Heller's method, all shift changes occur at the end of a block of consecutive weeks for each shift. At each of these shift changes, there is always at least one day off. Thus, there are no shift changes during a working period. It was proved in Chapter 3 that the Three Stage Method generates all schedules with no mid-working period shift changes for each rotation. Thus for any desired combination of days-off periods, the Three Stage Method, using Stages 2 and 3,

generates all schedules that are generated by Heller's procedure. In addition, other schedules with different structures are generated. The Three Stage Method does have the drawback that the set, or sets, of days-off periods must be determined, either by Stage 1 or by some other means. However, most employees have preferences for sets of days-off periods and by specifying the set of days-off periods, time is saved by not generating schedules with undesirable combinations of days-off periods. Also, it is likely that several different structures of schedules would be suitable. Thus, the Three Stage Method would be more suitable than Heller's method.

Heller acknowledged that the structure of the schedules generated by his method would not be suitable for all scheduling problems. In order to obtain a somewhat wider range of schedules, he suggested that the block of consecutive weeks be split up and interchanged with blocks of other shifts but did not provide a procedure to accomplish this. One method would be to split the shifts up before using his method but this would require running the entire method for each splitting-up arrangement and executing his second stage once for each possible ordering of the blocks of weeks. A better approach would be to use the days-off periods of a schedule to be split up as input to Stage 2 of the Three Stage Method. By using Algorithm 3.3 for Stage 3, this use of the Three Stage Method would generate all schedules that would be generated by exhaustive

use of Heller's approach. Thus, for those situations where a Heller type of schedule is desired, the Three Stage Method can be used in conjunction with Heller's method to produce a wider range of schedules. Clearly this results in longer execution times than with using only Heller's method but increases the chance of obtaining an acceptable schedule.

To illustrate the use of the two methods, an example with 9 groups was considered. A condition was imposed such that any days-off period longer than two days should include both Saturday and Sunday. Heller's method produced 6 schedules, none of which satisfied the above condition. His method was then modified to eliminate the objective of minimizing the maximum number of consecutive working weekends from the second stage objective function. Using the modified procedure, 14 more schedules were produced; one of these satisfied the above condition. The CPU time was 3.25 seconds. The days-off periods from this schedule were used in Stage 2 of the Three Stage Method; 7 rotations were generated. Algorithm 3.3 was used on these 7 rotations and generated 18 schedules, one of which was the schedule generated earlier by Heller's method; the schedule is shown in Figure 42. The other 17 feasible schedules would not be generated by Heller's method. The CPU time used for these two stages was 0.7 seconds.[33] Of the three constraints that were used for Stage 3 in the last section, only the first,

----

[33] It was shown in section 4.2.1 that the CPU time for Stage 1 ranged from 0.062 to 0.074 seconds for problems of the same size.

Week #	Mon	Tue	Wed	Thu	Fri	Sat	Sun
1	*	N	N	N	N	N	N
2	N	*	*	N	N	N	N
3	N	N	N	*	*	*	*
4	A	A	A	A	A	A	*
5	*	A	A	A	A	A	A
6	A	*	*	A	A	A	A
7	*	*	D	D	D	D	D
8	D	D	*	*	D	D	D
9	D	D	D	D	*	*	*

Legend: N - Night shift

D - Day shift

A - Afternoon shift

* - Day off

FIGURE 42 - Schedule Generated by Heller's and Three Stage Method

that there must be Day shift following Afternoon shift before Night shift, was used; this constraint was implicitly applied with Heller's method because the order of the shifts for his second stage was Nights, Afternoons and Days.

Heller's method was also applied to several of the example cases of the previous section. It was found that for those cases with less than 9 groups, no schedules were produced. Because of the restrictiveness of the structure imposed by Heller's method, the likelihood of being able to split the shifts into blocks of whole weeks decreases as the length of the schedule decreases. Thus, in addition to Heller's method cnly being suitable for scheduling problems with a specific type of schedule desired, it is not suitable for cases with few grcups of employees.

4.4 Summary of Results

In this chapter, the usefulness of the Three Stage Method described in Chapter 3 was demonstrated. This general method for generating cyclic shift schedules was shown to be able to solve realistic shift scheduling problems. It is described as a general method because the method can be applied to any cyclic shift scheduling problem, not just a limited subset of problems. However, in order for the method to solve problems in a realistic length of time, two approaches were used in conjunction with each other. The first consisted of setting an objective function; the objective function used was to minimize the number of shift

changes during a working period. This enabled much faster algorithms to be used in the third stage. The second approach was to add several constraints, to both Stages 2 and 3, in order to reduce the set of allowable schedules. Because of the reduction in execution time obtained by using these constraints, a procedure of applying as many constraints as possible and then relaxing them until suitable schedules are found was recommended. It was observed that the complexity of the scheduling problem depends more on the constraints being applied than on the length of the schedule desired.

The Three Stage Method is significantly different from existing cyclic shift scheduling methods. Many did not address the problem of specifying the shifts worked; for those that did it was largely an ad hoc, trial and error process. Several methods were suitable for specific shift scheduling situations. Only one other method, that developed by Heller et al., used an enumerative technique. This latter procedure imposed certain restrictions on the type of schedules which limited its applicability. Furthermore, it was found to be not suitable for developing schedules consisting of relatively few weeks, ie. few groups of employees. However, according to Rutenfranz et al. [43], the psychological well-being of workers improves by working shorter shift schedules. Thus the Three Stage Method, which does not have these limitations, is better suited for a wider range of problems. To further demonstrate the

generality of this new method, it was shown that besides generating those schedules produced by Heller's method, it also could be used in conjunction with his method to produce similar schedules.

A successful application and implementation of the Three Stage Method is discussed in the next chapter to further demonstrate the usefulness of the method.

CHAPTER 5 An Application of the Three Stage Shift Scheduling Method

5.1 Introduction to the Problem

This chapter outlines the application of the Three Stage Method described in Chapter 3 to shift scheduling in the Edmonton Police Department. It was demonstrated in Chapter 4 that if appropriate constraints were applied, the Three Stage Method could be used, even for a relatively large number of groups (15). This method, with either Algorithm 3.3 or 3.5 for the third stage, generates shift schedules in a reasonable amount of computing time. A real-life example will be described in Section 5.2 illustrating these points. However, first some background history of shift scheduling in the Edmonton Police Department (EPD) will be stated briefly here.

Until 1976, shift schedules for use in the EPD were generated entirely by a manual, trial and error process, often performed by one supervisor. In 1976 the Management of the Police Department wished to alter the manpower allocation and the Edmonton Police Association[33] as well expressed a desire for change, namely a different form of shift schedule. Both of these changes necessitated a new

----

[33]The Edmonton Police Association is the recognized bargaining unit for police officers up to and including the rank of Sergeant-Major.

shift schedule. In designing the new schedule, Management, in consultation with the Association, wished to explore as wide a range of schedules as possible. The method of Heller et al. [22, 23, 24] was initially used but the schedules produced were not considered suitable. A manual approach, a forerunner of the Three Stage Method, was developed but it suffered from the following two drawbacks. It was a slow process and, as with any manual procedure, only a small range of schedules could ever be generated. This manual procedure was used and a new schedule was adopted in February, 1977. However, it was clear that a better, automated method was necessary for future shift scheduling.

5.2 The Example

In the fall of 1977, Management and the Association decided to again change the shift schedule; the Three Stage Method described in Chapter 3 was used for generating the schedules. Consistent with the method, the approach that was adopted was to add as many constraints as possible and to later relax them if suitable schedules were not produced. These constraints were devised by the President of the Association after much consultation with the members. As shift changes during a working period were seen as undesirable, Algorithm 3.4, which attempted to minimize the number of such changes, was utilized for the third stage. Algorithm 3.5 had not yet been developed.

The initial constraints that were used, in addition to meeting the manpower allocation and following the shift precedence requirement[34], were:

- There is to be no more than one working period of seven days in length in the schedule and no working periods of longer duration.
- The minimum length of an acceptable working period is four days.
- At least two days in succession must be worked on any given shift[35].
- Only Day shift is to precede the days-off period preceding Night shift.[36]
- In the working periods preceding and following weekends off, only Nights or Afternoons are to be worked.[37]

As the Association appeared to be indifferent to having two three-day weekends off or having one four-day weekend and one two-day weekend off, two sets of days-off periods were generated. This was accomplished by applying Stage 1 twice. One set was generated with the utility of two 3-day

----

[34] Day or Afternoons may not directly precede Night shifts and Afternoons may not precede Days shift, without intervening days off.

[35] The three shifts used in the Edmonton Police Department, Nights, Days and Afternoons, are defined to consist of those tours of duty whose major portion falls into the intervals Midnight - 8 AM, 8 AM - 4 PM, 4 PM - Midnight, respectively. This definition is used because there are several starting times for each shift.

[36] If Afternoons preceded the days-off period, the length of the days-off period in hours would be less than with Days.

[37] The rationale behind this constraint is that because when working Day shift during the weekend, the evenings are free for socializing, the weekends off are of greater benefit when working Nights and Afternoons.

weekends being greater than the utility of one 4-day and one 2-day weekend. The other set was generated with the the utility function reflecting the opposite relation. These two sets of days-off periods necessitated two complete applications of the method for the second and third stages. As only five unique schedules were generated, the number of duplicate schedules did not present an inconvenience. These are given in Appendix II. However, neither the Chief of Police nor the Association President were sufficiently satisfied with the five. They then proposed relaxing the first constraint permitting two working periods seven days long providing they were not successive periods. Within a few hours, they had copies of three new schedules that met the revised constraints. Of these, the one shown in Figure 43 was selected by both the Association and the Chief of Police; the other schedules are given in Appendix II. The principal factors that set this schedule apart were:

- There were no shift changes in the middle of a working period.
- The two seven-day working periods were on Day shift.
- Of the fourteen days on Night shift, obtained from the manpower allocation, half are the "Midnight" or "graveyard" tour from Midnight to 8 AM[38]. With this schedule, no more than four Midnights need be worked consecutively.
- There are two 'passes' through Nights, Afternoons and

---

[38] The other Night tour is from 9 PM to 5 AM.

Week #	Mon	Tue	Wed	Thu	Fri	Sat	Sun
1	*	*	D	D	D	D	D
2	D	D	*	*	N	N	N
3	N	*	*	A	A	A	A
4	A	*	*	D	D	D	D
5	D	D	D	*	*	N	N
6	N	N	N	N	*	*	*
7	*	N	N	N	N	*	*
8	A	A	A	A	A	A	*
9	*	A	A	A	A	A	A

Legend: N - Night shift

D - Day shift

A - Afternoon shift

* - Day off

FIGURE 43 - Schedule Selected by EPD

Days, a traditional ordering of the shifts. Also, this meant that there was not an unacceptably long period of time on any one shift.

This schedule was implemented on February 27, 1978 and is still in operation.

5.3 Summary

The above example demonstrates that the Three Stage Method can be a great asset in designing suitable shift schedules in a real scheduling situation. Considerable time is saved over a manual approach and because a wider range of possible schedules are explored by the method, there is more likelihood of producing desirable schedules. Due to changing conditions, such as different manpower deployment requirements or altered preferences due to new personnel, the shift schedule will need to be changed from time to time. The Three Stage Method facilitates this task.

CHAPTER 6 Future Research

6.1 Future Research in Shift Scheduling

The major emphasis of the research for this thesis was on the development of procedures for assigning shifts to working days. Two extensions of this work are outlined below. The Three Stage Method developed in this thesis along with several other methods described in the literature utilize objective functions based on the preferences of employees. These functions are usually assumed to be known and procedures for developing them are not described. In section 6.1.3, one possible direction for developing these functions will be discussed.

6.1.1 Algorithms for Generating Schedules with Other Objective Functions

Algorithm 3.2 (described in Section 3.4) is a general algorithm that, given a cyclic schedule of days worked and days off, will generate all feasible shift assignments. Three similar, but much faster-executing algorithms (Alg. 3.3 - 3.5) were developed that adopted the objective of minimizing the number of shift changes during working periods. However, there are many shift scheduling situations where this objective is not appropriate. According to Knauth

and Rutenfranz [31] and others [41, 43], for many round-the-clock operations, a rapid rotation of shifts (ie. frequent shift changes) is desirable from a physiological point of view. Thus one direction of future research is to develop faster algorithms for Stage 3 that would adopt this and other objectives.

6.1.2 Algorithms for Non-Cyclic Schedules

The Three Stage Method described in Chapter 3 generates cyclic schedules. However, as demonstrated in the literature review, there is a need for shift assignment procedures when employees are scheduled individually. Miller [37] developed an enumerative procedure for generating individual schedules but only suggested a trial and error approach for the assignment of shifts[38]. Two possibilities exist for assigning shifts in conjunction with Miller's method. The first is to incorporate shift objectives and constraints for each individual into the penalty function formulation. Such a change may, however, result in an algorithm which would require an unreasonable length of time to solve shift assignment problems. The second possibility is to extend the methodology of the Stage 3 algorithms of the Three Stage Method and develop a procedure to assign shifts to the schedules produced by Miller's existing method. Such a procedure could be used to assign shifts to any individual

---

[38] Miller's method was developed for fixed shift applications.

schedule, regardless of the method that produced the schedule.

6.1.3 Incorporating Multi-Attribute Preferences

The third area of future research to be outlined is the incorporation of the wishes of employees concerning their schedules. The Three Stage Method, along with others described in Chapter 2 such as the method of Bodin [10], assumed the existence of objective functions that reflect the relative utility of various features and attributes of shift schedules. One possibility for developing these functions is to apply multi-attribute utility theory.[39] This theory can be described as the theory of developing utility functions over alternatives having several attributes or effectiveness measures that cannot be quantitatively related. Multi-attribute utility theory has been applied to as diverse projects as deciding on the location of a new Mexico City Airport [28] and developing a utility function for response times of fire-fighting equipment to be used for resource allocation [27]. In the former application, the relative utilities of six attributes, including investment costs, safety and noise pollution, were measured for several government officials. These utility functions were used to formulate an overall objective function. In the latter application, the utility of various combinations of response

[39] A comprehensive text on this subject is Keeney and Raiffa [29].

levels for different equipment types were measured for one senior fire department official. These utilities were used to formulate an objective function that could be used in a resource allocation model for the deployment of fire-fighting equipment. For an application to the generation of shift schedules, it could be possible to combine several attributes, such as frequency of shift changes, maximum number of consecutive working weekends and average length of working periods, in a manner reflecting the utilities and preferences of the employees. The use of multi-attribute utility theory would be a time-consuming task, if one attempted to elicit the preferences of all employees being schedules. However, if a small number of representatives can be identified from the group as in Chapter 5, then the application of this theory could be practical.

Three directions of future research in shift scheduling have been suggested. Although beyond the scope of this dissertation, they would extend the applicability of several existing methods, including the Three Stage Method described in Chapter 3.

BIBLIOGRAPHY

[1] W. J. Abernathy, N. Baloff, J. C. Hershey and S. Wandel, "A three-stage manpower planning and scheduling model - a service sector example", Operations Research, vol. 22 no. 3, May/June 1973, pp. 693-711.

Forecasting of manpower needs and the allocation of personnel are dealt with but without shift scheduling.

[2] Aho, A. V., Hopcroft, J. E. and Ullman, J. D., The Design and Analysis of Computer Algorithms , Addison-Wesley, 1974.

This text is used as a reference on the subject of computational time complexity.

[3] H. Ahuja and R. Sheppard, "Computerized Nurse Scheduling", Industrial Engineering, vol. 7 no. 10, October 1975, pp.24-29.

A procedure is described that includes both the allocation stage as well as the shift schedule generation stage. The rotation of the days off in the schedules is cyclical but the assignment of shifts is not. This method has been implemented in a hospital in Newfoundland.

[4] Akerstedt, T., "Interindividual Differences in Adjustment to Shift Work", Proceedings of 6th Congress of the International Ergonomics Association, Univ. of Maryland, 1976.

The differing capabilities for adjustment to shift work by employees is investigated through the use of a quetionnaire to a group of steel workers in Sweden. The most significant result of the study was that the well-being of older workers, particularly those who had been exposed to shift work for a long time, was worse than the well-being of younger workers.

[5] K. R. Baker, "Scheduling full-time and part-time staff to meet cyclical requirements", Operational Research Quarterly, vol.25, March 1974.

Baker is concerned only with the allocation of a mix of full-time and part-time employees where consecutive days off are ensured. No method of generating schedules is given.

[6] , "Workforce Allocation in Cyclic Scheduling Problems: A Survey", Operational Research Quarterly, Vol. 27, No. 1, 1976.

An overview of manpower allocation strategies with a brief discussion of some approaches to shift scheduling.

[7] Baker, K.R. and Magazine, M. J., "Workforce Scheduling with Cyclic Demands and Day-off Constraints", Management Science, Vol. 24, No. 2, October 1977, pp. 161-167.

Algorithms are presented that are guaranteed to devise a shift schedule that meets specified manpower requirements with a minimum size of workforce and satisfies various constraints on the formation of days-off periods. With the exception of the constraints, an example of which is to have every other weekend off and to have four days off every two weeks, employees' preferences are not incorporated. The type of demand for manpower is limited toeh case where N are required on weekdays and n, possibly equal to N, on weekends.

[8] B. Bennett and R. Potts, "Rotating Roster for a Transit System", Transportation Science, vol. 2 no. 1, February 1968, pp. 15-34.

Rotations are generated by a computerized heuristic algorithm that, in addition to satisfying all constraints, attempts to best satisfy the preferences of the employees by forming as many long recreation clusters as possible and spreading them

evenly throughout the schedule. The former objective is accomplished in a stepwise manner by first generating as many four day recreation clusters as possible, then three day, then two day, where spreading equations are used to achieve the latter. Shifts are assigned by a heuristic procedure that minimizes overtime and finds 'good' schedules with respect to two cost functions that are approximate evaluations of employees' preferences. However, there was no actual measurement of these preferences.

[9] Berztiss, A. T., DATA STRUCTURES, Theory and Practice , Academic Press, 1971.

This text is used as a reference for tree structures and tree-based algorithms.

[10] L. D. Bodin, "Towards a General Model for Manpower Scheduling", Urban Analysis, vol.1 1973, Part 1 pp. 191-207, Part 2 pp. 223-245.

In part 1, Bodin explains why manpower scheduling is important and outlines a general framework in which to imbed scheduling methods. In part 2, he develops the model in detail and shows how four existing procedures are subsets of the general case. The four methods studied are due to Altman, Bodin and others for the New York City Department of Sanitation (described in detail in this paper), Bennett and Potts for the Municipal Tramways Trust, Adelaide, South Australia (see [4]), Heller for the St. Louis Police Department (see [13], [14] and [15]) and Monroe [25] for baggage handlers at an airport. Bodin's model is subdivided into four submodels: allocation of the workforce, grouping of days off into recreation clusters, formation of a rotation and assignment of shifts.

[11] E. S. Buffa, M. J. Cosgrove and B. J. Luce, "An Integrated Work Shift Scheduling System", Decision Sciences, vol. 7 no. 4, October 1976, pp. 620-630.

This deals primarily with the forecasting of

telephone operator requirements, developing the actual shifts, and the allocation of operators to these shifts. At the end of the paper, the actual shift scheduling problem is mentioned, briefly describing an algorithm developed by Luce [21]. This algorithm appears quite heuristic and very structured but does allow ordinal preferences of shifts for each day of the week to be input for each operator. The shifts are assigned on a seniority basis and are not cyclical. It also specifies lunch periods and rest periods (ie. coffee breaks) for each shift.

[12] R. W. Butterworth and G. T. Howard, "A Method of Determining Highway Patrol Manning Schedules", ORSA 44th National Meeting, November 1973.

This paper deals principally with manpower allocation.

[13] J. G. Church, "Sure Staff: A Computerized Staff Scheduling System for Telephone Business Offices", Management Science, vol. 20 no. 4, December 1973, pp. 708-720.

This paper is concerned with the forecasting of demand for a group of telephone workers and the allocation of these employees but not with actual shift schedules.

[14] Chaiken,J. M. and Dormont, P., Patrol Car Allocation Model: User's Manual, The New York City Rand Institute, 1975.

This manual describes the M/M/n queueing model used in PCAM in addition to the methods of operation of the package. The model can either be used as a descriptive or as a prescriptive package. In descriptive mode, it will estimate various performance measures, such as probability of encountering a queue, average delay time and average number of availbale servers, given hourly workload and the allocation of patrol cars. Alternatively, using the prescriptive mode, it will specify the

number of patrol cars needed by shift to attain specified peformance levels.

[15] A. F. Dalley, "A Systematic Approach to Shift Scheduling", RCMP Gazette, vol. 38 no. 11, 1977.

This describes a manual method of allocating and scheduling manpower that was designed for relatively small RCMP detachments and although the allocation stage is reasonably effective, the shift scheduling stage is an inadequate ad hoc procedure. This paper is included because it is a good example of a technique that deals primarily with efficiency of allocation with little recognition of the human factors connected with shift work.

[16] Feller, W., An Introduction to Probability Theory and Its Applications, Second Edition, J. Wiley, 1960.

A general reference on probability from which the approximation of n! was taken.

[17] Sister M. A. Frances, "Implementing a program of cyclical scheduling of nursing personnel", Hospitals, Journal of the American Hospital Association, vol. 40, July 1966.

This describes an application of heuristic procedures based on the work of Monroe and discusses the involvement of both the systems engineering staff and the nurses in this successful implementation.

[18] Guha, D. and Browne, J., "Optimal Scheduling of Tours and Days Off", presented at ORSA Workshop on Automated Techniques for Scheduling of Vehicle Operators for Urban Public Transportation Services, April 1975, Chicago.

This paper describes Guha's algorithm for allocating workers based on hourly requirements of

each day and the algorithm of Tibrewala, Phillippe and Browne[31] for determining the number of workers needed satisfying the constraint of two consecutive days off. They then discuss some ad hoc procedures, including those that include paid holidays in order to ensure feasibility, that could be used to generate the shift schedule.

[19] D. Harris ,"Staffing Requirements ", Hospitals, Journal of the American Hospitals Association, vol. 44, April 16, 1970, pp. 64-70.

This paper does not deal with generating shift schedules but with the allocation of different categories of nursing staff to shifts.

[20] W. C. Healy, "Shift scheduling made easy", Factory, vol. 117 no. 10, October 1959.

An entirely manual, trial and error approach but this paper appears to be the beginning of the search for systematic procedures for devising shift shcedules.

[21] N. B. Heller, "Proportional Rotational Schedules", Ph. D. Thesis (1969), University of Pennsylvania.

[22] N. B. Heller, J. T. McEwen and W. W. Stenzel, "Police Manpower Scheduling", Board of Police Commissioners, St. Louis Police Department, St. Louis, Missouri, 1972.

[23] N. B. Heller, J. T. McEwen and W. W. Stenzel, "Computerized Scheduling of Police Manpower", Nelson B. Heller & Associates, St. Louis, Missouri, March 1973.

The above documents describe a method developed by Heller and its implementation in some areas of the St. Louis Police Department. It is an enumerative method that partitions the days off into clusters of acceptable lengths and generates all feasible cycles (ie. rotations). The objective function used to select alternatives is a crude,

lexicographic ordering based on frequencies of twelve different weekends, the ranking of which is at the discretion of the user. Each shift is scheduled separately, thus necessitating each shift to be allocated an even number of employee-weeks, and then all shifts are combined by selecting each individual shift's rotation in such a manner (by explicit enumeration) as to maximize the same lexicographic objective function as above. The schedules generated by this method all have the property of having a number of weeks of the first shift, followed by a number of weeks on the second shift, and so on until a number of weeks of the last shift. Thus for schedules of many weeks duration, there are long periods on each shift and long periods of time between occurrences of a particular shift. This slow rotation can be altered by breaking up a shift into several 'shifts', but this is a trial and error approach and Heller's method rarely generates feasible schedules if the length of a shift (ie. the number of weeks) is small. Because of this, Heller's method is not well suited to cases where the number of groups being scheduled is small.

[24] J. P. Howell, "Cyclical scheduling of nursing personnel", Hospitals, Journal of the American Hospital Association, vol. 40 January 1966, pp. 71-85.

This article describes an appilcation of heuristic procedures based on the work of Monroe [25].

[25] International Mathematical and Statistical Libraries (IMSL), Reference Manual, Sixth Edition, Houston, 1977.

This is a reference manual for the linear programming subroutine used in this thesis.

[26] R. L. Keeney, "A Utility Function for the Response Times of Engines and Ladders to Fires", Urban Analysis, vol. 1, 1973, pp. 209-222.

This paper describes an application of Keeney's theoretical work (see [17] and [18]) to the fire department response problem where the preferences of one Deputy Chief are measured and utilized.

[27] R. L. Keeney, "A Decision Analysis with Multiple Objectives: the Mexico City Airport", Bell Journal of Economics and Management Science, vol. 4, 1973, pp. 101-117.

This describes the procedures developed by Keeney and used by the Mexican government in determining a thirty year policy on airport usage and construction in the vivinity of Mexico City.

[28] Keeney, R. L. and Raiffa, H., <u>Decisions with Multiple Objectives: Preferences and Value Tradeoffs</u> , Wiley, 1976.

A comprehensive text on multi-attribute decision theory.

[29] Knauth, P., Rohmert, W. and Rutenfranz, J., "Systematic Selection of Shift Plans for Continuous Production with the Aid of Work-Physiological Criteria", Proceedings of 6th Congress of the International Ergonomics Association, Univ. of Maryland, 1976.

Of principal concern to the authors is the ratio of working days to the days off. For Based on physiological effects, they determine acceptable values for this ratio and give examples of shift rotations that achieve this value. However, no procedure was given for determining these shift schedules. Furthermore, they were only concerned with an equal manpower requirement around the clock.

[30] Knauth, P., and Rutenfranz, J., "Experimental shift work studies of permanent and rapidly rotating shift systems", International Archives of Occupational and Environmental Health, Vol. 37 No. 2, 1976, pp. 125-137.

The physiological effect of night shift are studied. The conclusion reached that it is better to work a rapidly rotating shift system than one with permanent night shift. The study was bsed on an analysis of a small number of subject, sometimes living under artificial conditions.

[31] Kolesar, P. J., Rider, K. L., Crabill, T. B. and Walker, W. E.,"A Queuing-Linear Programming Approach to Scheduling Police Patrol Cars", Operations Research, Vol. 23, No. 6, 1975, pp1045-1062.

A two-stage method utilizing queuing theory and integer linear programming, is used to determine the number of patrol cars to be deployed on each tour in order to meet specified performance standards. The first stage is an M/M/n queuing model with time-dependent parameters that is solved numerically. The second stgae is an integer program in which the decision variables are the number of patrol cars working each tour and the times of their meal breaks. The program's constraints are determined by the output of the queuing model.

[32] Kregeloh, H. and Miodrag, M., "Automated Formation of Staff Schedules and Duty Rosters", (Hamburger Hochbahn Ag.), presented at ORSA Workshop on Automated Techniques for Scheduling of Vehicle Operators for Urban Public Transportation Services, April 1975, Chicago.

This describes procedures for creating runs, assigning operators and generating duty rosters. This last stage is by a manual heuristic that schedules six days on followed by at least two days off. All work periods commence on a week day and in order to provide as long recreation periods as possible, each working period starts with a late evening shift and rotates ahead and end with an early morning shift.

[33] B. J. Luce, "A Shift Scheduling Algorithm", ORSA 44th Meeting, November 1973.

A heuristic procedure is developed to schedule employees individual. This method is described in [6].

[34] C. Maier-Rothe and H. R. Wolfe, "Cyclical Scheduling and Allocation of Nursing Staff", Socio-Economic Planning Sciences, vol. 7, 1973, pp. 471-487.

The method as outlined in this paper develops feasible alternate schedules and allows the nurses to choose one. Rather than using an enumerative technique, this method uses an integer program that includes constraints as to the minimum and maximum lenghts of working periods.

[35] McCartney, B. McKee, C. D. Cady, "Nurse Staffing Systems ", Hospitals, Journal of the American Hospital Association, vol. 44, November 16 1970, pp. 102-105.

This paper does not deal with generating shift schedules but with the allocation of different categories of nursing staff to shifts.

[36] H. E. Miller, "Relationships Between Automated Scheduling Techniques for Nurses and Public Transportation Vehicle Operators", presented at ORSA Workshop on Automated Techniques for Scheduling of Vehicle Operators for Urban Public Transportation Services, April 1975, Chicago. Troy, New York.

This paper describes a method for generating non-cyclical, ie. individual, schedules for a small group of people. In generating the schedules, Miller minimizes an objective function that is composed of the sum of two classes of penalty costs; the first class deals with the violation of non-binding staffing level constraints while the second class consists of violation of non-binding schedule pattern constraints. Miller defines costs of violating these constrints, weights for each employee for these violations and a weight that is a measure of how good or bad an employee's past schedules have been vis-a-vis their preferences but

he does not state how these weigths and costs should be calculated. Furthermore, he does not address the problem of scheduling multiple shifts except as a future extension.

[37] G. Monroe, "Scheduling manpower for service operations", Industrial Engineering, August 1970, pp. 10-17.

This paper presents heuristic approaches to shift scheduling that can be used on a manual basis and are essentially trial and error processes to generate a feasible schedule.

[38] A. Morrish and A. O'Connor, "Cyclic Scheduling", Hospitals, Journal of the American Hospital Association, vol. 44, February 16 1970, pp. 66-71.

This paper describes a computerized heuristic method to generate schedules where the days off pattern is cyclic but the shift assignment is not. They also allow adjustment of the manpower allocation and use the rule that, if there is to be understaffing, then this should occur on day shift.

[39] Oginski, A., Kozlakowska-Swigon, L. and Pokorski, J., "Diurnal and Seasonal Variations in Industrial Fatigue of Shift Workers", Proceedings of 6th Congress of the International Ergonomics Association, Univ. of Maryland, 1976.

Using questionnaires measuring subjective feelings of fatigue and objective measures such as heart rate, the authors found significant differences in the effects of the various shifts of Polish steel workers. As well, these effects showed a seasonal variation. Because of this, the authors recommended short periods of shift rotation.

[40] Reinberg A., Vieux, N., Laporte, A., Ghata, J. and Migraine,C., "Rapid Adjustment of Circadian Rhythms in Shift Workers of an Oil Refinery", Proceedings of 6th Congress of the International Ergonomics

Association, Univ. of Maryland, 1976.

After measuring physiological variables in a small group of workers, they found a capability to adjust well to rapidly rotating shifts. They conclude that on this basis, in additional to perceived social benefits, that the rapidly rotating scheme is preferable to a slower rotation.

[41] M. Rothstein, "Scheduling manpower with mathematical programming", Industrial Engineering, vol. 4 no. 4, April 1972, pp. 29-33.

This describes a procedure that given an allocation of manpower, it calculates recreation clusters by a linear program that maximizes days off pairs (ie. consecutive).

[42] Rutenfranz, J., Knauth, P. and Colquhoun, W. P., "Hours of Work and Shift Work", Ergonomics, Vol. 19, No. 3, 1976, pp. 331-340.

Of principal concern was the length of the working day and the incidence of night shift. They concluded that for most occupations, working days beyond eight hours in length should not be tolerated and they recommended a rapidly rotating shift system with at least twenty-four hours off after a night shift.

[43] L. D. Smith, "The Application of an Interactive Algorithm to Develop Cyclical Rotational Schedules for Nursing Personnel", INFOR, Canadian Journal of Operational Research and Information Processing, vol. 14 no. 1, February 1976, pp. 53-70.

This is an example of using an interactive approach in generating schedules that allows the user to alter the original manpower allocation in order to produce a feasible schedule. There is no provision for employees' preferences and the schedules are generated in an ad hoc manner, both in terms of developing a rotation of days off and

assigning the shifts worked.

[44] R. Tibrewala, D. Philippe and J. Browne, "Optimal Scheduling of Two Consecutive Idle Periods", Management Science, vol. 19 no. 1, September 1972, pp. 71-75.

This paper develops an efficient algorithm that produces the minimum allocation of personnel that simultaneously meets minimum manpower constraints and ensures that recreation periods can consist of two consecutive days. If this latter property is a constraint in the shift scheduling problem, this procedure can be used to derive an alternate allocation if the original one is not feasible.

[45] D. M. Warner and J. Prawda, "A Mathematical Programming Model for Scheduling Nursing Personnel in a Hospital", Management Science, vol. 19 no. 4, December Part I 1972, pp. 411-422.

This paper describes an elaborate manpower allocation scheme but does not address the problem of generating shift schedules. The model, similar to production scheduling models as it incorporates labour and personnel shortage costs in addition to expected demands, is designed to be run twice a week with a horizon of three or four days, necessitating a very flexible shift scheduling arrangement.

APPENDIX I

Program Listings

of the

Three Stage Method

```
C
C          PROGRAM CALLS ZX3LP FROM IMSL
C          IS INTENDED TO PERFORM LP FOR SELECTION OF
C          MOST PREFERRED DAYS OFF PERIODS
C
C       MAX NUMBER OF GROUPS IS 30
C
        DIMENSION A(10,30),B(10),C(30),PSOL(30),
       * DSOL(10),RW(150),IW(35),B7(7),DSOL8(8)
        EQUIVALENCE (B(2),B7),(DSOL,DSOL8)
C
        READ(5,901) NGRP,MINGRP
901     FORMAT(2I3)
C
C       A IS READ IN BY COLUMN
C
           DO 110 J=1,NGRP
           READ(5,902) (A(I,J),I=2,8)
902        FORMAT(7F2.0)
C
C          SET UP INEQUALITY CONSTRAINT
C          ( MIN # OF GRPS )
C
           A(1,J)=-1.0
110        CONTINUE
        READ(5,902) B7
        B(1)=-MINGRP
        READ(5,903) (C(J),J=1,NGRP)
903     FORMAT(20F4.0)
C
        CALL ZX3LP(A,10,B,C,NGRP,1,7,S,PSOL,DSOL,RW,
       *   IW,IER)
C
        IF (IER.NE.0) GO TO 990
C
120     CONTINUE
        WRITE(6,911) S,(PSOL(J),J=1,NGRP)
911     FORMAT('-      OPTIMAL VALUE OF OBJECTIVE ',
       * 'FUNCTION',F10.3,/,
       * '0     PRIMAI SOLUTION',//,(8F7.3))
        WRITE(6,912) DSOL8
912     FORMAT('0      DUAL SOLUTION',(8F7.3),//)
        STOP
990     CONTINUE
        WRITE(6,913) IER
913     FORMAT('-***** ERROR *****      IER IS',I5)
        IF (IER.EQ.70) GO TO 120
        STOP
        END
```

```
BEGIN
COMMENT  THIS IS STAGE 2;
INTEGER START,KLIM,DLIM,Z,FST,FDJ,I,NOGOOD,IDJ;
INTEGER FLAG,NZERO,I1,DIF,SUM,NWORK,DTEMP,FDJ1;
COMMENT  GET # OF UNIQUE DOP'S AND TOTAL # OF
      DOP'S AND TOTAL # OF WORKING DAYS;
GET(5,"3I3",DLIM,KLIM,NWORK);
  BEGIN
  INTEGER ARRAY DOFF,ALLOW (1::7);
  INTEGER ARRAY NDOP,FDAY,LDAY (1::DLIM);
  INTEGER ARRAY ORDER,LENWP (1::KLIM);
  INTEGER ARRAY DOP (1::DLIM,1::7);
  INTEGER ARRAY FOLLOWS (1::DLIM,1::DLIM);
  INTEGER ARRAY MAXLEN,NMLEN (2::9);
  STRING (10) ARRAY MSG (0::3);

  PROCEDURE OUTPUT(INTEGER ARRAY ORDER (*);
    INTEGER VALUE PARM,KTHIS);
    BEGIN
    IF PARM = 1 THEN
      BEGIN
      PUT(8,"H ,A10",MSG(1));
      FOR I:=1 UNTIL KTHIS DO
        PUTON(8,"A1,I2,A1,I2,A1,2X","(",LENWP(I),
          ",",ORDER(I),")");
      END
    ELSE
      BEGIN
      PUT(7,"H ,A10",MSG(PARM));
      FOR I:=1 UNTIL KTHIS DO
        PUTON(7,"A1,I2,A1,I2,A1,2X","(",LENWP(I),
          ",",ORDER(I),")");
      IF PARM = 3 THEN PUTON(7,"2X,A6,I3",
        "SUM = ",SUM);
      END;
    END OUTPUT;

  PROCEDURE BUILD(INTEGER VALUE ROOT,KTHIS;
    INTEGER ARRAY DOFFP,NDOPP,NUMLENP(*));
    BEGIN
    INTEGER ARRAY NDOPPREM (1::DLIM);
    INTEGER ARRAY REMDOFFP (1::7);
    INTEGER ARRAY NUMLEN(2::9);
    INTEGER NEWKTHIS,NEWDJ,LIM,DJ,DIF;
    FOR I:=2 UNTIL 9 DO NUMLEN(I):=NUMLENP(I);
    FOR DJ:=1 UNTIL DLIM DO
      BEGIN
      IF FOLLOWS(ROOT,DJ)¬=0 AND NDOPP(DJ)>0 THEN
        BEGIN
        COMMENT ADJUST # OF DOP'S REMAINING;
        FOR I:=1 UNTIL DLIM DO NDOPPREM(I):=
          NDOPP(I);
        NDOPPREM(DJ):=NDOPPREM(DJ)-1;
        COMMENT THE ORDER OF THIS CYCLE;
```

```
ORDER(KTHIS):=DJ;
NEWKTHIS:=KTHIS+1;
COMMENT UPDATE # OF DAYS OFF REM;
FOR I:=1 UNTIL 7 DO REMDOFFP(I):=DOFFP(I)
  -DOP(DJ,I);
COMMENT THIS PART TEST TO SEE IF
  THERE IS A CUTOFF;
FST:=FDAY(START);
LDJ:=Z:=LDAY(DJ);
FDJ:=FDAY(DJ);
FDJ1:= IF FDJ=1 THEN 7 ELSE FDJ-1;
DIF:=FDJ1-LDAY(ROOT);
IF DIF < 0 THEN DIF:=DIF+7;
IF DIF<=1 THEN DIF:=DIF+7;
LENWP(KTHIS):=DIF;
NUMLEN(DIF):=NUMLEN(DIF)+1;
IF NUMLEN(DIF)<=MAXLEN(DIF) THEN
  BEGIN
  IF NEWKTHIS<KLIM THEN BUILD(DJ,
    NEWKTHIS,REMDOFFP,NDOPPREM,NUMLEN)
  ELSE
    BEGIN
    COMMENT ONLY ONE DOP LEFT;
    NEWDJ:=1;
    WHILE NDOPPREM(NEWDJ)=0 DO NEWDJ
      :=NEWDJ+1;
    ORDER(NEWKTHIS):=NEWDJ;
    IF FOLLOWS(DJ,NEWDJ)¬=0 AND
    FOLLOWS(NEWDJ,START)¬=0  THEN
      BEGIN
      COMMENT DETERMINE LENGTHS OF
       WP'S AND TOTAL # OF WORKING
       DAYS.  3 <= LENGTH OF WP <= 9
       ASSUMED. TOTAL MUST = NWORK.
       ;
      SUM:=0;
      COMMENT GET LENGTH OF FIRST
        AND LAST WP;
      FDJ:=FDAY(NEWDJ);
      DIF:=FDJ - LDJ;
      DIF:= IF DIF <= 2 THEN DIF+6
        ELSE DIF-1;
      IF DIF <= 0 THEN DIF:=DIF+7;
      LENWP(NEWKTHIS):=DIF;
      NUMLEN(DIF):=NUMLEN(DIF)+1;
      DTEMP:=DIF;
      DIF:=FDAY(START) - LDAY(NEWDJ);
      DIF:= IF DIF <= 2 THEN DIF+6
        ELSE DIF-1;
      IF DIF <= 0 THEN DIF:=DIF+7;
      LENWP(1):=DIF;
      NUMLEN(DIF):=NUMLEN(DIF)+1;
      FOR I:=1 UNTIL NEWKTHIS DO
        SUM:=SUM+LENWP(I);
```

```
              IF SUM=NWORK AND NUMLEN(DIF) <=
                MAXLEN(DIF) AND NUMLEN(DTEMP)
                 <= MAXLEN(DTEMP) THEN
                OUTPUT(ORDER,1,NEWKTHIS) ELSE
                  OUTPUT(ORDER,3,NEWKTHIS);
                END
              ELSE OUTPUT(ORDER,0,NEWKTHIS);
            END;
          END;
        END;
      END;
    END BUILD;

  MSG(0):="NO CYCLE  ";
  MSG(1):="  CYCLE   ";
  MSG(2):="CUTOFF #2 ";
  MSG(3):="CUTOFF #3 ";
  FOR DJ:=1 UNTIL DLIM DO
    BEGIN
    GET(5,"I3",DOP(DJ,1));
    FOR I:=2 UNTIL 7 DO GETON(5,"I3",DOP(DJ,I));
    END;
  GET(5,"I3",DOFF(1));
  FOR I:=2 UNTIL 7 DO GETON (5,"I3",DOFF(I));
  GET(5,"I3",NDOP(1));
  FOR I:=2 UNTIL DLIM DO GETON (5,"I3",NDOP(I));
  GET(5,"I3",ALLOW(1));
  FOR I:=2 UNTIL 7 DO GETON (5,"I3",ALLOW(I));
  FOR DJ:=1 UNTIL DLIM DO
    BEGIN
    GET(5,"I3",FOLLOWS(DJ,1));
    FOR I:=2 UNTIL DLIM DO GETON(5,"I3",
      FOLLOWS(DJ,I));
    END;
  FOR I:= 1 UNTIL DLIM DO GET(5,"2I3",FDAY(I),
    LDAY(I));
  START:=1;
  WHILE NDOP(START)=0 DO START:=START+1;
  FOR DJ:=1 UNTIL DLIM DO
    BEGIN
    IF NDOP(DJ)<NDOP(START) AND NDOP(DJ)>0 THEN
       START:=DJ;
    END;
  ORDER(1):=START;
  NDOP(START):=NDOP(START)-1;
  FOR I:=1 UNTIL 7 DO DOFF(I):=DOFF(I)
    -DOP(START,I);
  COMMENT READ MAX # OF WP'S FOR EACH LENGTH;
  GET(5,"3X,I3",MAXLEN(2));
  FOR I:=3 UNTIL 9 DO GETON(5,"I3",MAXLEN(I));
  FOR I:=2 UNTIL 9 DO NMLEN(I):=0;
  BUILD(START,2,DOFF,NDOP,NMLEN);
  END;
END.
```

```
BEGIN
COMMENT THIS PROGRAM DEVELOPS FEASIBLE SCHEDULES,
   COMPLETE WITH SHIFT ASSIGNMENTS,
    FROM A ROTATION;
COMMENT THIS IS ALG 3.3 OF STAGE 3
  ;
INTEGER NDOP,NSHIFTS,NWKS,TOTDAYS,FDAY,NDIF,SCHEDNO
  ,NCON,AFT,NROT,OK;
NCON:=3;
SCHEDNO:=1;
GET(5,"4I3",NDIF,NDOP,NSHIFTS,NWKS);
  BEGIN
  INTEGER ARRAY START,LEN,NEXTWP,WPWEEK,DOP,
    SELECTED,ASSIGNED (1::NDOP);
  INTEGER ARRAY FSTWP,DAYS (1::7);
  INTEGER ARRAY LDOP (1::NDIF);
  INTEGER ARRAY SCHEDULE (1::NWKS,1::7);
  INTEGER ARRAY ALLTOTDAYS,MINDAYS (1::NSHIFTS);
  INTEGER ARRAY ALLDAYS (1::NSHIFTS,1::7);
  INTEGER IDAY,IWK,IDOP,WPI,IDOP1,I1;
  INTEGER ARRAY CONSTR(1::NCON);
  INTEGER ARRAY ALLOW(1::NSHIFTS,1::NDOP);
  INTEGER ARRAY OKAY(1::NSHIFTS,1::NDIF);


  PROCEDURE SELECT(INTEGER VALUE SHIFT,TOTDAYS,
  LASTWPI; INTEGER ARRAY START,LEN,DAYS,FSTWP,
  NEXTWP,WPWEEK,SELECTED,ASSIGNED(*));
    BEGIN
    COMMENT THIS PROCEDURE DETERMINES THE WP TO BE
       SELECTED FOR THIS SHIFT;
    INTEGER I,I1,FST,WPI,STRTWP;
    INTEGER ARRAY TASSIGNED(1::NDOP);
    COMMENT MAKE COPY;
    FOR K:=1 UNTIL NDOP DO TASSIGNED(K):=
      ASSIGNED(K);
    COMMENT SCAN FOR CASE DAYS(J) < DAYS(J+1);
    I:=1;
    I1:=2;
    WHILE I <=7 AND DAYS(I) >= DAYS(I1) DO
      BEGIN
      I:=I+1;
      I1:= IF I1=7 THEN 1 ELSE I1+1;
      END;
    IF I=8 THEN
      BEGIN
      COMMENT NO CASE, THEREFORE ATTEMPT ALL WP'S;
      FOR STRTWP:=1 UNTIL 7 DO
        BEGIN
        IF FSTWP(STRTWP) ¬= 0 THEN
          BEGIN
          WPI:=FST:=FSTWP(STRTWP);
          COMMENT ENSURE WPI ALLOWED FOR THIS
            SHIFT;
```

```
              IF (SELECTED(WPI) = 0) AND
              (TASSIGNED(WPI) = 0) AND  (CONSTR(2)=0
              AND CONSTR(3)=0 OR ALLOW(SHIFT,WPI)=1)
              THEN
                BEGIN
                TASSIGNED(WPI):=1;
                ASSIGN(WPI,SHIFT,TOTDAYS,LASTWPI,START,
                LEN,DAYS, FSTWP,NEXTWP,WPWEEK,SELECTED,
                TASSIGNED);
                END;
              WPI:=NEXTWP(WPI);
              WHILE WPI ¬= FST DO
                BEGIN
                COMMENT  ENSURE WPI ALLOWED FOR THIS
                  SHIFT;
                IF (SELECTED(WPI) = 0) AND
                (TASSIGNED(WPI) = 0) AND  (CONSTR(2)=0
                AND CONSTR(3)=0 OR ALLOW(SHIFT,WPI)=1)
                THEN
                  BEGIN
                  TASSIGNED(WPI):=1;
                  ASSIGN(WPI,SHIFT,TOTDAYS,LASTWPI,
                  START,LEN,DAYS,FSTWP,NEXTWP,WPWEEK,
                  SELECTED,TASSIGNED);
                  END;
                WPI:=NEXTWP(WPI);
                END;
              END;
            END;
          END
        ELSE
          BEGIN
          COMMENT THERE IS A CASE, THEREFORE WE MUST
            USE A WP STARTING WITH I1. WE ONLY NEED
            TO CONSIDER THE FIRST CASE BECAUSE IF IT
            DOESN'T FIT, IT WILL NOT BE POSSIBLE TO
            DEVELOP A FEASIBLE SCHEDULE. STARTING ON
            ANOTHER CASE WON'T CHANGE IT;
          STRTWP:=I1;
          IF FSTWP(STRTWP) ¬= 0 THEN
            BEGIN
            COMMENT TRY ALL WP'S STARTING ON THIS DAY;
            WPI:=FST:=FSTWP(STRTWP);
            COMMENT  ENSURE WPI ALLOWED FOR THIS SHIFT;
            IF (SELECTED(WPI) = 0) AND
            (TASSIGNED(WPI) = 0) AND  (CONSTR(2)=0 AND
            CONSTR(3)=0 OR ALLOW(SHIFT,WPI)=1) THEN
              BEGIN
              TASSIGNED(WPI):=1;
              ASSIGN(WPI,SHIFT,TOTDAYS,LASTWPI,START,
              LEN,DAYS, FSTWP,NEXTWP,WPWEEK,SELECTED,
              TASSIGNED);
              END;
            WPI:=NEXTWP(WPI);
```

```
          WHILE WPI ¬= FST DO
            BEGIN
            COMMENT  ENSURE WPI ALLOWED FOR THIS
              SHIFT;
            IF (SELECTED(WPI) = 0) AND
            (TASSIGNED(WPI) = 0) AND  (CONSTR(2)=0
            AND CONSTR(3)=0 OR ALLOW(SHIFT,WPI)=1)
            THEN
              BEGIN
              TASSIGNED(WPI):=1;
              ASSIGN(WPI,SHIFT,TOTDAYS,LASTWPI,START,
              LEN,DAYS,FSTWP,NEXTWP,WPWEEK,SELECTED,
              TASSIGNED);
              END;
            WPI:=NEXTWP(WPI);
            END;
          END;
        END;
      END SELECT;


    PROCEDURE ASSIGN(INTEGER VALUE WPI,SHIFT,TOTDAYS,
      LASTWPI; INTEGER ARRAY START,LEN,DAYS,FSTWP,
      NEXTWP,WPWEEK,SELECTED,ASSIGNED (*));
      BEGIN
      COMMENT THIS PROCEDURE ASSIGNS THE SHIFT TO
        THE WP;
      INTEGER STRTWP,WPLEN,IWEEK,IDAY,TEMP,I,SUCCESS,
      ISH,NEWWPLEN ,WPLENJ,WPLENLIM,TSHIFT,TTOTDAYS,
      ILEN,FLAG;
      INTEGER ARRAY TSTART,TLEN,TNEXTWP,TWPWEEK,
        TSELECTED (1::NDOP);
      INTEGER ARRAY TDAYS,TFSTWP (1::7);


      PROCEDURE ASSIGNREM;
        BEGIN
        COMMENT ASSIGN LAST SHIFT TO ALL
              UNSELECTED WP'S;


        PROCEDURE PAD;
          BEGIN
          INTEGER WPI1;
          IDAY:=TSTART(WPI);
          IWEEK:=TWPWEEK(WPI);
          FOR J:=1 UNTIL TLEN(WPI) DO
            BEGIN
            SCHEDULE(IWEEK,IDAY):=TSHIFT;
            IDAY:=IDAY+1;
            IF IDAY > 7 THEN
              BEGIN
              IDAY:=1;
              IWEEK:= IF IWEEK = NWKS THEN 1 ELSE
```

```
              IWEEK+1;
            END;
          END;
        IF CONSTR(1)=1 AND TSHIFT=AFT THEN
          BEGIN
          COMMENT CHECK IF NEXT SHIFT A NIGHT
            (SHIFT 1);
          IDAY:=IDAY+LDOP(DOP(WPI));
          WHILE IDAY > 7 DO
            BEGIN
            IDAY:=IDAY-7;
            IWEEK:= IF IWEEK=NWKS THEN 1 ELSE
              IWEEK+1;
            END;
          COMMENT WPI1 IS THE NEXT WP. TSTART(WPI1)
            ¬= IDAY IF IT HAS ALREADY BEEN ASSIGNED
            AT LEAST ONCE. IF IT HAS NOT BEEN
            ASSIGNED THEN IT WILL BE THIS LAST
            SHIFT WHICH IS OKAY;
          WPI1:= IF WPI=NDOP THEN 1 ELSE WPI+1;
          IF SCHEDULE(IWEEK,IDAY) = 1 AND
          (TSTART(WPI1) ¬= IDAY OR TWPWEEK(WPI1)
          ¬= IWEEK) THEN SUCCESS:=0;
          END;
        END PAD;


      SUCCESS:=1;
      FOR I:=1 UNTIL 7 DO
        BEGIN
        IF TFSTWP(I) ¬= 0 THEN
          BEGIN
          WPI:=TFSTWP(I);
          PAD;
          TEMP:=WPI;
          WPI:=TNEXTWP(TEMP);
          WHILE WPI ¬= TEMP AND SUCCESS = 1 DO
            BEGIN
            PAD;
            WPI:=TNEXTWP(WPI);
            END;
          END;
        END;
      IF SUCCESS = 1 THEN
        BEGIN
        COMMENT OUTPUT THE SCHEDULE;
        PUT(6,"H-,A10,I4","SCHEDULE #",SCHEDNO);
        SCHEDNO:=SCHEDNO+1;
        PUT(6,"H0,A4,7X,A19","WK #",
          "M  T  W  T  F  S  S");
        FOR I:=1 UNTIL NWKS DO
          BEGIN
          PUT(6,"H0,I3,6X",I);
          FOR J:=1 UNTIL 7 DO PUTON(6,"I3",
```

```
                SCHEDULE(I,J));
            END;
          END;
        END ASSIGNREM;

      COMMENT MAKE COPIES OF ARRAYS;
      FOR I:=1 UNTIL NDOP DO
        BEGIN
        TSTART(I):=START(I);
        TLEN(I):=LEN(I);
        TNEXTWP(I):=NEXTWP(I);
        TWPWEEK(I):=WPWEEK(I);
        TSELECTED(I):=SELECTED(I);
        END;
      TSELECTED(WPI):=1;

      TTOTDAYS:=TOTDAYS;
      IDAY:=TSTART(WPI);
      IWEEK:=TWPWEEK(WPI);
      ILEN:=0;
      FOR I:=1 UNTIL 7 DO TDAYS(I):=DAYS(I);
      WPLEN:=LEN(WPI);
      WPLENLIM:= IF TOTDAYS<WPLEN THEN TOTDAYS
        ELSE WPLEN;
      FLAG:=0;
      WPLENJ:=WPLEN;

      COMMENT RESET ARRAYS;
      STRTWP:=TSTART(WPI):=START(WPI);
      TLEN(WPI):=LEN(WPI);
      TWPWEEK(WPI):=WPWEEK(WPI);
      FOR I:=1 UNTIL NDOP DO TNEXTWP(I):=NEXTWP(I);
      FOR I:=1 UNTIL 7 DO  TFSTWP(I):=FSTWP(I);
      COMMENT INITIALIZE;
      TSHIFT:=SHIFT;
      WHILE ILEN<WPLENJ AND TDAYS(IDAY) > 0 DO
        BEGIN
        ILEN:=ILEN+1;
        SCHEDULE(IWEEK,IDAY):=TSHIFT;
        TDAYS(IDAY):=TDAYS(IDAY)-1;
        TTOTDAYS:=TTOTDAYS-1;
        IDAY:=IDAY+1;
        IF IDAY > 7 THEN
          BEGIN
          IDAY:=1;
          IWEEK:= IF IWEEK=NWKS THEN 1 ELSE IWEEK+1;
          END;
        END;
      COMMENT ENSURE BIG LOOP IS NOT REPEATED;
      IF TDAYS(IDAY)=0 THEN FLAG:=1;
      COMMENT ENSURE MIN # OF CONSEC DAYS OF THIS
        TSHIFT SATISFIED. COULD ALSO HAVE A MAX #
        CONSTRAINT; COMPLETE WP'S ARE ATTACHED;
      IF ILEN >= MINDAYS(TSHIFT) THEN
```

```
        BEGIN
        COMMENT THIS WP ACCEPTABLE;
        SUCCESS:=1;
        COMMENT GET RID OF WP FROM LIST;
        TEMP:=TNEXTWP(WPI);
        WHILE TNEXTWP(TEMP) ¬= WPI DO TEMP:=
          TNEXTWP(TEMP);
        TNEXTWP(TEMP):=TNEXTWP(WPI);
        COMMENT SET INDICATOR THAT WPI IS ASSIGNED;
        TSTART(WPI):=0;
        IF TFSTWP(STRTWP) = WPI THEN
          BEGIN
          TFSTWP(STRTWP):= IF TEMP=WPI THEN 0
            ELSE TEMP;
          END;
        IF ILEN < WPLEN THEN
          BEGIN
          COMMENT ENSURE NEW (SHORTENED) WP
            ACCEPTABLE;
          NEWWPLEN:=WPLEN-ILEN;
          ISH:=TSHIFT+1;
          COMMENT NEW WP MUST BE >= MIN # OF CONSEC
            DAYS FOR AT LEAST ONE OF THE REMAINING
            SHIFTS;
          WHILE ISH <= NSHIFTS AND NEWWPLEN <
            MINDAYS(ISH) DO ISH:=ISH+1;
          IF ISH > NSHIFTS THEN SUCCESS:=0 ELSE
            BEGIN
            COMMENT ADD THIS NEW (SHORTENED) WP;
            IF TFSTWP(IDAY) = 0 THEN
              BEGIN
              TFSTWP(IDAY):=WPI;
              TNEXTWP(WPI):=WPI;
              END
            ELSE
              BEGIN
              TNEXTWP(WPI):=TEMP:=TFSTWP(IDAY);
              WHILE TNEXTWP(TEMP) ¬= TFSTWP(IDAY) DO
                TEMP:=TNEXTWP(TEMP);
              TNEXTWP(TEMP):=WPI;
              END;
            TSTART(WPI):=IDAY;
            TLEN(WPI):=WPLEN-ILEN;
            TWPWEEK(WPI):=IWEEK;
            END;
          END;

        IF SUCCESS = 1 THEN
          BEGIN
          IF TTOTDAYS = 0 THEN
            BEGIN
            COMMENT END FOR THIS SHIFT;
            COMMENT RESET LASTWPI;
            LASTWPI:=0;
```

```
            TSHIFT:=TSHIFT+1;
            IF TSHIFT < NSHIFTS THEN
              BEGIN
              FOR K:=1 UNTIL NDOP DO
                BEGIN
                TSELECTED(K):=0;
                ASSIGNED(K):=0;
                END;
              FOR K:=1 UNTIL 7 DO TDAYS(K):=
                ALLDAYS(TSHIFT,K);
              TTOTDAYS:=ALLTOTDAYS(TSHIFT);
              SELECT(TSHIFT,TTOTDAYS,LASTWPI,TSTART,
                TLEN,TDAYS, TFSTWP,TNEXTWP,TWPWEEK,
                TSELECTED,ASSIGNED);
              END
            ELSE ASSIGNREM;
            END
          ELSE SELECT(TSHIFT,TTOTDAYS,LASTWPI,TSTART,
            TLEN,TDAYS, TFSTWP,TNEXTWP,TWPWEEK,
            TSELECTED,ASSIGNED);
          END;
        END;
      END ASSIGN;


  COMMENT BACK IN MAIN PROGRAM;

  COMMENT FIRST DAY OF FIRST DOP;
  GET(5,"I3",FDAY);
  COMMENT LENGTH OF DOP'S;
  GET(5,"I3",LDOP(1));
  FOR I:=2 UNTIL NDIF DO GETON(5,"I3",LDOP(I));
  COMMENT GET WORKING DAYS BY SHIFT AND DAY;
  FOR I:=1 UNTIL NSHIFTS DO
    BEGIN
    GET(5,"I3",ALLDAYS(I,1));
    TOTDAYS:=ALLDAYS(I,1);
    FOR J:=2 UNTIL 7 DO
      BEGIN
      GETON(5,"I3",ALLDAYS(I,J));
      TOTDAYS:=TOTDAYS+ALLDAYS(I,J);
      END;
    ALLTOTDAYS(I):=TOTDAYS;
    END;
  TOTDAYS:=ALLTOTDAYS(1);
  FOR J:=1 UNTIL 7 DO DAYS(J):=ALLDAYS(1,J);
  COMMENT GET MIN # OF CONSEC DAYS FOR EACH SHIFT;
  GET(5,"I3",MINDAYS(1));
  FOR I:=2 UNTIL NSHIFTS DO GETON(5,"I3",
    MINDAYS(I));
  COMMENT GET WHICH CONSTRAINTS IN FORCE;
  GET(5,"I3",CONSTR(1));
  FOR I:=2 UNTIL NCON DO GETON(5,"I3",CONSTR(I));
  COMMENT GET ALLOW MATRIX. ALLOW(I,J) = 1 IF
```

```
    SHIFT I ALLOWED BEFORE AND/OR AFTER JTH UNIQUE
    DOP;
  FOR I:=1 UNTIL NSHIFTS DO
    BEGIN
    GET(5,"I3",OKAY(I,1));
    FOR J:=2 UNTIL NDIF DO GETON(5,"I3",OKAY(I,J));
    END;
  COMMENT GET WHICH SHIFT IS AFTERNOONS;
  GET(5,"I3",AFT);
  COMMENT TRY ALL ROTATIONS;
  COMMENT GET NUMBER OF ROTATIONS;
  GET(5,"I3",NROT);

  COMMENT DO THIS BLOCK FOR EACH ROTATION;

  FOR IROT:=1 UNTIL NROT DO
    BEGIN
    PUT(6,"H-,A10,I3","ROTATION #",IROT);
    GET(5,"12X,I2,1X,I2",LEN(1),DOP(1));
    FOR I:=2 UNTIL NDOP DO GETON(5,"3X,I2,1X,I2",
      LEN(I),DOP(I));

    IF CONSTR(2)=1 OR CONSTR(3)=1 THEN
      BEGIN
      COMMENT SET UP ALLOW MATRIX;
      FOR I:=1 UNTIL NDOP DO
        BEGIN
        IDOP:=DOP(I);
        COMMENT CALC PREVIOUS DOP;
        I1:= IF I=1 THEN NDOP ELSE I-1;
        IDOP1:=DOP(I1);
        FOR J:=1 UNTIL NSHIFTS DO
          BEGIN
          IF (CONSTR(2)=1 AND OKAY(J,IDOP)=0) OR
             (CONSTR(3)=1 AND OKAY(J,IDOP1)=0)
             THEN OK:=0 ELSE OK:=1;
          ALLOW(J,I):=OK;
          END;
        END;
      END;

    COMMENT BUILD SCHEDULE MATRIX AND DATA
      STRUCTURE;
    IDOP:=1;
    IDAY:=FDAY;
    IWK:=1;
    FOR I:=1 UNTIL 7 DO FSTWP(I):=0;
    WPI:=2;
    WHILE IDOP <= NDOP DO
      BEGIN
      FOR I:=1 UNTIL LDOP(DOP(IDOP)) DO
        BEGIN
        SCHEDULE(IWK,IDAY):=0;
        IDAY:=IDAY+1;
```

```
          IF IDAY > 7 THEN
            BEGIN
            IDAY:=1;
            IWK:= IF IWK=NWKS THEN 1 ELSE IWK+1;
            END;
          END;
        COMMENT SET UP WP;
        START(WPI):=IDAY;
        WPWEEK(WPI):=IWK;
        IF FSTWP(IDAY) = 0 THEN FSTWP(IDAY):=
          NEXTWP(WPI):=WPI
        ELSE
          BEGIN
          NEXTWP(WPI):=NEXTWP(FSTWP(IDAY));
          NEXTWP(FSTWP(IDAY)):=WPI;
          END;
        FOR I:=1 UNTIL LEN(WPI) DO
          BEGIN
          COMMENT SET SCHEDULE ENTRY TO A
            NONEXISTENT SHIFT;
          SCHEDULE(IWK,IDAY):=9;
          IDAY:=IDAY+1;
          IF IDAY > 7 THEN
            BEGIN
            IDAY:=1;
            IWK:= IF IWK=NWKS THEN 1 ELSE IWK+1;
            END;
          END;
        IDOP:=IDOP+1;
        WPI:= IF WPI=NDOP THEN 1 ELSE WPI+1;
        END;

      FOR K:=1 UNTIL NDOP DO
        BEGIN
        SELECTED(K):=0;
        ASSIGNED(K):=0;
        END;

      COMMENT START THE PROCESS;
      SELECT(1,TOTDAYS,0,START,LEN,DAYS,FSTWP,NEXTWP,
        WPWEEK,SELECTED,ASSIGNED);
      END;
    END;
  END.
```

APPENDIX II

Sample Schedules for the Edmonton Police Department

Week #	Mon	Tue	Wed	Thu	Fri	Sat	Sun
1	*	*	D	D	D	D	D
2	D	*	*	D	D	D	D
3	D	D	*	*	N	N	N
4	N	D	D	*	*	N	N
5	N	N	N	N	*	*	*
6	*	N	N	N	N	*	*
7	A	A	A	A	A	A	*
8	*	A	A	A	A	A	A
9	A	*	*	A	A	A	A

Legend: N - Night Shift

D - Day Shift

A - Afternoon Shift

* - Day off

Week #	Mon	Tue	Wed	Thu	Fri	Sat	Sun
1	*	*	D	D	D	D	D
2	D	*	*	A	A	A	A
3	A	*	*	D	D	D	D
4	D	D	*	*	N	N	N
5	N	D	D	*	*	N	N
6	N	N	N	N	*	*	*
7	*	N	N	N	N	*	*
8	A	A	A	A	A	A	*
9	*	A	A	A	A	A	A

Legend: N - Night Shift

D - Day Shift

A - Afternoon Shift

* - Day off

Week #	Mon	Tue	Wed	Thu	Fri	Sat	Sun
1	*	*	D	D	D	D	D
2	D	*	*	N	N	A	A
3	A	*	*	D	D	D	D
4	D	D	*	*	N	N	N
5	N	N	N	*	*	N	N
6	N	N	N	N	*	*	*
7	*	A	A	A	A	*	*
8	A	A	A	A	A	A	*
9	*	D	D	A	A	A	A

Legend: N - Night Shift

D - Day Shift

A - Afternoon Shift

* - Day off

Week #	Mon	Tue	Wed	Thu	Fri	Sat	Sun
1	*	*	D	D	D	D	D
2	D	*	*	N	N	A	A
3	A	*	*	D	D	D	D
4	D	D	*	*	N	N	N
5	N	N	N	*	*	N	N
6	N	N	N	N	*	*	*
7	A	A	A	A	A	*	*
8	*	A	A	A	A	A	*
9	*	D	D	A	A	A	A

Legend: N - Night Shift

D - Day Shift

A - Afternoon Shift

* - Day off

Week #	Mon	Tue	Wed	Thu	Fri	Sat	Sun
1	*	*	D	D	D	D	D
2	D	*	*	N	N	D	D
3	D	D	*	*	N	N	N
4	N	N	N	*	*	N	N
5	N	N	N	N	*	*	*
6	A	A	A	A	A	*	*
7	*	A	A	A	A	A	*
8	*	D	D	D	D	A	A
9	A	*	*	A	A	A	A

Legend: N - Night Shift

D - Day Shift

A - Afternoon Shift

* - Day off

Week #	Mon	Tue	Wed	Thu	Fri	Sat	Sun
1	*	*	D	D	D	D	D
2	D	D	*	*	N	N	N
3	N	D	D	*	*	N	N
4	N	N	N	N	*	*	*
5	*	N	N	N	N	*	*
6	A	A	A	A	A	A	*
7	*	A	A	A	A	A	A
8	A	*	*	D	D	D	D
9	D	*	*	A	A	A	A

Legend: N - Night Shift

D - Day Shift

A - Afternoon Shift

* - Day off

Week #	Mon	Tue	Wed	Thu	Fri	Sat	Sun
1	*	*	D	D	D	D	D
2	D	D	*	*	N	N	N
3	D	D	D	*	*	N	N
4	N	N	N	N	*	*	*
5	N	N	N	N	N	*	*
6	*	A	A	A	A	A	*
7	*	A	A	A	A	A	A
8	A	*	*	A	A	A	A
9	A	*	*	D	D	D	D

Legend: N - Night Shift

D - Day Shift

A - Afternoon Shift

* - Day off